日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００２年１２月　６日

出願番号
Application Number: 特願２００２－３８３０８３

[ST. 10/C]: ［ＪＰ２００２－３８３０８３］

出願人
Applicant(s): 加藤　宝一

２００３年１２月　４日

特許庁長官
Commissioner,
Japan Patent Office

今井康

出証番号　出証特２００３－３１００２８７

【書類名】　特許願

【整理番号】　KTMEPS21

【提出日】　平成14年12月 5日

【あて先】　特許庁長官　殿

【発明者】

【住所又は居所】　大阪府大阪市鶴見区鶴見５丁目６番９号３０１室

【氏名】　加藤　宝一

【特許出願人】

【住所又は居所】　大阪府大阪市鶴見区鶴見５丁目６番９号３０１室

【氏名又は名称】　加藤　宝一

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図　面　1

【物件名】　要約書　1

【書類名】　明細書

【発明の名称】　簡易式入出力プログラミングシステム

【特許請求の範囲】

【請求項１】　信号入出力及びアナログ入出力及び操作キー入力及び通信送受及び表示出力のプログラミング作製のため、特定言語へのマッピングを行う，選択型コマンドジェネレーター。

【請求項２】　請求項１で作製された言語の多重処理用簡易コンパイラー。

【請求項３】　請求項２のコンパイラーでコンパイルしたオブジェクトを実行させるコントローラー。

【発明の詳細な説明】

【０００１】

信号入出力処理、アナログ入出力処理、通信送受処理、操作キー入力処理、表示出力処理を初心者が平易に設計し得る旨のビジュアルかつ記述選択型のコマンドジェネレーターをパソコンにて実行させ、コーディング手法を用いずとも、システム運用を遂行させるものである。

【発明の属する技術分野】

本発明は、産業用計測機、制御機のプログラミング手法の平易化に関する。

【０００２】

【従来の技術】

コンピューター一般における従来のプログラミング手法は、コーディング形式によるもの。産業用入出力処理装置としては線画及び記号を付加して行くピジュアル形式のラダー図によるものが主たるものである。コーディング形式は近年、より一層拡充された各種コンパイラーがそれぞれの機能を競っており、優劣をつけがたいものがあるが、概して、元来コンパイラーは関数演算、ファイルマネジメント、画面入出力等諸機能を時系列で処理するを旨とし、時には多重処理用ＯＳを持ったものもある。しかし、多点のＩ／Ｏを処理するにはコーディング文字数が多くなり、不便である。一方ラダー図の方はハードウェア技術者にとって設計回路に近く、ラダー図シーケンシャル処理は先頭ステップから末尾ステップまでを一瞬にして実行をサイクリックに繰り返すものでＩ／Ｏ専一機としては高速

であるが、時系列のヒストリー組立てには熟練度を要する。つまり統合する意志的なものは常にユーザー側にあり、個人的な個性、能力差によってはシステム状態数の増大にともない、幾何級数的に複雑度が増大する。更にプログラムの流れは一次元的に流れるので、多重処理用ＯＳを持たないものは、フラッグやメモリスイッチやメモリーのスケジューラーを製作者が作り出して多重処理を構築する。

【０００３】

【発明が解決しようとする課題】

以上のように述べた従来のプログラミング製作システムでは初心技術者では状態ステップが多ければ多いほと開発出来なくなる（あるいは長期間になる）ので、この問題を一挙に解決するものである。

【０００４】

問題解決に３通りあり、第一に，コーディングミスをさけるためにも、ネーミング検証方式にて処理し、記述選択方式にて、コーディングさせない事。

第二に，多重処理と各処理の連携の統合、規制をコマンドジェネレーターもしくは簡易コンパイラーで統合、規制すべきものである事。

第三に，多重処理を意識せずに各々のタスクについてはあくまでもフローチャートの流れを反影出来る形式でなければならない。極論すれば命令書の参照なしに製作出来る事。音声会話形式のロボットティーチング指令の将来へつなげる事。

【課題を解決するための手段】

本発明は上記目的を達成するため、コントローラーにおける多重ＯＳ用のカーネル（核）を必要最小なものとして、各々のタスクの現在状況を示す、フラッグとイベント発生を待つエントリーアドレス、その時のスタックポインター、各々タスクが持つタイマーカウンター等々をサーチするモニター、タイマーチック割り込みにおけるタイマーのカウントダウン処理に留める。

【０００５】

簡易コンパイラー言語に入出力内部信号待ちと監視タイマーと組み合わせて１パックにした、１ステップ動作と動作の完全遂行をチェックするマクロ命令を製

作する。

更に、必要箇所で遅延及びイベントウェイトを行うマクロ命令を作る事により、フローチャート形式にする。

更に一次元的に一瞬にしてサーチするタスク行を並べ、それぞれのタスク内の動作をコマンド列にしフローチャート形式で書ける完全二次元的構造にする。

更に二次元表現のため、従来のコマンドステートメントの単位が改行ベースなのに対して、本コンパイラーのコマンドの区切りにスペースを使用、右端で改行となるまでコマンドステートメントを横に展開出来るようにする。その事により、タスク行コマンド列の二次元表現とする。コメントは；から；まで又は改行までをコメントとしているからどの箇所でも挿入出来る。

更に他のコンパイラー同様各種関数を用意する。

【０００６】

コマンドジェネレーターとして、各種処理のメニュー選択をさせる。

更にタスク列を必ず宣言するようにコマンドジェネレーターで製作はガイダンスに基づき、ミスを防ぐ事。

更にコマンドジェネレーターで入出力内部信号の１点１点をネーミングさせ、注釈も表示させ、ミスを防ぐ事。

更にコマンドジェネレーターを使用せず、直接簡易コンパイラーを修正して、熟練度に応じて更なる応用性を拡充出来るようにすべくエディターも内包する事。

更にコマンドジェネレーターはパソコンとコントローラーをＲＳ２３２Ｃで接続して．プログラム転送及びデバッグ機能を有する事。

【０００７】

【発明実施の形態】

以下、本発明の実施の形態を図１～図５０に基づいて説明する。

図１の如く、パソコンＰＣＡＴ互換機ＷＩＮＤＯＷＳにて、コマンドジェネレーターで生成されたマクロ言語を簡易コンパイラーでコンパイルし、コントローラーへオグジェクトをＲＳ２３２Ｃで転送して実行するものである。

更に簡易コンパイラーで生成されたマクロ言語はテキスト形式なので、コマン

ドジェネレーターなしでも、コンパイルして転送できるものである。したがってコマンドジェネレーター、簡易コンパイラー、コントローラーと分けて説明する。

【コマンドジェネレーター】

【０００８】

コマンドジェネレーターは図１の如く、コマンドジェネレーター、それに併う独自開発のエディター、デバッガーを内包する。コマンドジェネレーターはコマンドメニューにて実行する。エディターは編集メニュー、デバッガーはデバッグメニューにて実行する。

エディターは、コピー、切り取り、貼り付け、検索、置換機能を持つ。

デバッガーは、デバッグモードでの実行（コントローラーＲＡＭ上で実行）、カーソルの前まで実行（ブレイクポイント）、ステップ等の機能を持つ。

【０００９】

図２は、コマンドジェネレーターの基本画面である。

図中、メニューの信号をクリックして入力、出力、内部信号のアサインを行う。これは端子番号にネーミングとコメントを入れる事で、システムの設計と確認を容易にするためである。マイクロソフトコンボボックスで信号ネームを、ツールチップテキストを用いてマウスの変化でコメントを表示するようにした。図３は入力信号、以下出力、内部信号は同様である。

【００１０】

図２におけるコマンドメニューで選択をした画面で代表的なものとして信号待ちを表示したものが図４である。タイマー付き入力待ち選択をして、入力信号待ちコマンドステートメントを生成させる。そして入力信号待ちに関するフローが図５である。これだけが簡易コンパイラーの一つのコマンドステートメントに相当する。

更に図２においてコマンドメニューでステップ動作を選択すれば図６の画面になり、ステップ動作の簡易コンパイラーへの生成は複数のコマンドステートメントとなる。図７中、＊１印はオプションによるもので、最小２最大５コマンドステートメントを生成する。このコマンドステートメント列の連続で大体のシーケ

ンス処理は完行される。図５の監視タイマーのセットは必ずしも必要でなく、その前にタイマーセットされており、信号チェックをひとまとめに監視するような場合、継続タイマーの選択も出来る。

【００１１】

純動作処理の他、制御データのパケット毎の搬送のため図８の如きシフトメモリー動作コマンドを用意して、回転テーブル、ライン搬送の実際的なデータフローに適応させる。動作イベントは入力、出力、内部信号の立ち上がり、立ち下がりで行う。

【００１２】

プロフラムステップ短縮する目的で縦×横のデータテーブルを用意し、インデックス的にデータを獲得する。図９、これにて例えば、テーブル化されたスピードコントロールデータ出力等の処理を簡易的に作成できる。

【００１３】

更に図１０の如くＡＮＤ，ＯＲゲートの選択が出来、信号判定と条件ジャンプ文を使用せずとも、適応したロジックを作製出来る。

【００１４】

更に図１１の如く４×４要素（図は４×３）のラダー論理図で信号のＡＮＤ，ＯＲを連結クリックと登録クリックで行い、従来の線画エディットに比して、快速に回路を作製出来る。

【００１５】

更に図１２の如くフリッカー選択で、アラーム等のフリッカーのコマンド列を選択可能にする。

【００１６】

更にレジスター関係の演算命令、関数、送受信マクロ、日付時計読み出し、アナログ入出力命令、タイマー命令等も記述型選択方式にてコーディング無しでコードを生成する。（図１３，図１４，図１５，図１６，図１７，図１８）

【００１７】

更にマルチタスク関連の命令は図１９の如くまとめ、簡単にタスク間の連結を行う。但しデータのやりとりは行わない。本システムのデータ、レジスター、信

号は全てパブリック状態にある。

【００１８】

更にコントローラー部のキー入力及びＬＣＤ出力の簡易化にともなう命令を図２０の如くまとめた。

【００１９】

更にジャンプ系列の命令、入出力内部信号の判定、セット命令も選択方式にて平易化した。（図２１，図２２）

【００２０】

コマンドジェネレーターは図１の如く、コマンドジェネレーター又は簡易コンパイラーの直接エディットを、用いて作製された言語をコンパイルしてオブジェクト生成後、コントローラーに送信して実行させるものである。デバッグモードまでは本コマンドジェネレーターの管理下にあるが．デバッグ完了後は、ＲＳ２３２Ｃを切り離し、コントローラ単独でフラッシュメモリー書き込み、作製されたプロフラムを実行するものである。

【００２１】

本コマンドジェネレーターに要するシステムはマイクロソフト各種ツール、コンポーネントである。

【簡易コンパイラー】

【００２２】

本コンパイラーは多重処理用に独自に開発したものである。

更に開発効率を高めるため、コンパイラー本来の特性、冗長度を持たした言語的な特性から、コンパクトに表現をする様、心がけたものである。そのため．ネモニックの工夫と命令の区切りをスペースにする等した。重点を入出力制御におき、一般のコンパイラーの様に演算式が自由でない。更に配列、インデックス等の機能は持たない。

更に構造上のクラス、メンバー等といった概念も持たない。但し、入出力の監視タイマー付取込み命令等１命令として特殊なものもある。（図５）

更にタスク命令として、生成、消滅、休止、リセット、ステータス等の命令を使用して、多重処理を実現させる。

【００２３】

本コンパイラーは“＝”と“＿”（アンダーライン）を中心として命令解析をツリー構造で行うものである。“＝”は右辺の式、又はデータを左辺へ演算、又は変換、又は直接代入するものである。

更に“＿”（アンダーライン）は左部のコマンドに対して、右部の詳細仕様に基づいたコマンドを実行する。命令にはタスク命令、ジャンプ命令、タイマー命令、Ａ／Ｄ、Ｄ／Ａ、送信、受信命令、演算命令、キー入力ＬＣＤ表示命令がある。

【００２４】

命令ネモニックの中でタスク命令以外でも“Ｓ”文字や“Ｗ”文字が含んでいる命令はタスクを切替（スイッチング）します。特に“Ｗ”含む命令は、その所（アドレス）で、とどまって、イベントをウェイトしている様態である。

更に“Ｓ”は送信を意味し送信完了まで、その所（アドレス）でとどまってウェイトする。

更に“％”は受信を意味し受信完了までウェイトする。

更に命令ネモニックの中に“？”文字を含んでいる命令は、判定ｔｒｕｅ（１）かｆａｌｓｅ（０）をチェックするものである。したがって次の命令は条件ジャンプが必ず必要となる。

【００２５】

各命令一覧表を図２３、図２４、図２５、図２６、図２７、図２８、図２９、図３０、に記す。便宜上タスクラベル名をＴａｓｋｎ、ラベル名をＰｃｌａｂｅｌ、レジスターＮＯを１０、２０、３０、データを９の複数桁又は最大値、入力信号をＩｓｉｇ、出力信号をＯｓｉｇ、内部信号名をＶｓｉｇ、アスキー文字列を”ＡＢＣ”、１６進文字列をｈ’３１、３２、３３、として表現する。

【コントローラー】

【００２６】

コントローラー仕様は図１如くである。更にコントローラーメモリー空間上、常駐モニター及び常駐サブルーチンを有する。

【００２７】

ＰＣから受信したコンパイル後のユーザーオブジェクトプログラムをそのままＲＡＭ上にデバッガーモードで走らせる事が出来る。

更に確認後、フラッシュモメリーへ書き込み処理を行い．ＰＣと切離してコントローラー単独で実用処理機となる。

更に、常駐のサービスプログラムを有する。ＰＣから転送されたユーザープログラムが開発者独自の自動プログラムとすれば、このサービスプログラムは手動処理的に提供しているものである。このサービスプログラムは主に３形態を有し、１つにはコントローラーのＰＣとの受信処理、フラッシュメモリー書き込み等のコントロール部と、１つには通信条件等のパラメータ設定、カレンダークロックの設定等の設定部と、１つには、Ｉ／Ｏ入出力、Ａ／Ｄ、Ｄ／Ａ入出力ＲＳ２３２Ｃ又はＲＳ４２２の送信受信、各々の手動チェック部とがある。そしてこれらの選択は初期画面から各々ツリー構造をなす画面推移で実現出来るようにしている。

【００２８】

コントローラーの仕様はＩ／Ｏフォトカプラー付き３２点／３２点、Ａ／Ｄ１２ビット×８ＣＨ　０－４．０９５Ｖ、Ｄ／Ａ１２ビット×２ＣＨ　０－５Ｖ、カレンダー時計、通信ＲＳ２３２Ｃ　１ＣＨ、ＲＳ４２２　１ＣＨ、１２８×６４ドット　ＬＣＤ（横１６桁×縦６桁アスキー文字表示器）、１２個のキー、５１２ＫＢフラッシュメモリー、５１２ＫＢスタティックＲＡＭ（バッテリー付）

ＣＰＵ　Ｚ８０１０ＭＨＺである。

【００２９】

内部メモリーとして本コマンドジェネレーター、本簡易コンパイラー、本コントローラー共通の設計仕様として、

ｖ００－ｖＦＦ、　２５６点の内部接点

ｉ００－ｉ９９、　１００個の１６ビットレジスター

ｄ００－ｄ９９、　１００個の１６ビット定数格納レジスター

ｔ００－ｔ９９、　１００個の１６ビットタイマー定数レジスター

ｒ００－ｒ９９、　１００個の３２ビット実数レジスター

ｃ００－ｃ９９、　１００個の８ビット文字レジスター

Ｔｍ０－Ｔｍ９、　１０個の１６ビット外部タイマー

各タスク１個１６ビット固有タイマー

最大タスク数１２０

システムサポート　内部接点　ｖｆｆ　１ｍｓデューティクロックサービス

内部接点　ｖｆｅ　１０ｍｓデューティクロックサービス

内部接点　ｖｆｄ　１ｓｅｃデューティクロックサービス

正整数レジスターｉ９９　ｍｓカウンター（０－９９９）

電源ＯＮのスタート時のデフォルト値として

実数レジスターｒ９９＝０．０

実数レジスターｒ９８＝１．０

実数レジスターｒ９７＝１０．０

実数レジスターｒ９６＝１００．０

実数レジスターｒ９５＝１０００．０

実数レジスターｒ９４＝４０９６．０

実数レジスターｒ９３＝３．１４１５９

実数レジスターｒ９２＝３６０．０

実数レジスターｒ９１＝２７３．０

【００３０】

１２キー入力のアスキーコードとしての対応図は図３１である。

【００３１】

図３２はメニュー画面、図３３はキーの説明でコントロール部の各々の画面は図３４～図３６。

設定部の各々の画面は図３７～図４４。チェックの各々の画面として図４５～図５０である。

【発明の効果】

上述した本発明のコマンドジェネレーター、簡易コンパイラー、コントローラーで製作工数が極端に減少し、デバッグミスがほとんどなく、初心技術者にも短時間のインストラクト後．適確に目的とする動作をさせる事が出来る効果がある。

【図面の簡単な説明】

【図１】

本簡易式入出力プログラミングシステムの構成図である。

【図２】

ＰＣ側コマンドジェネレーターの、メイン画面を示した図である。

【図３】

コマンドジェネレーターの、入力信号アサイン画面を示した図である。

【図４】

コマンドジェネレーターの、信号待ちコマンドの選択画面を示した図である。

【図５】

上記、信号待ちコマンド中、入力信号待ちのフローチャートを示した図である。

【図６】

コマンドジェネレーターの、ステップ動作コマンドの選択画面を示した図である。

【図７】

上記、ステップ動作複合コマンドのフローチャートを示した図である。

【図８】

コマンドジェネレーターの、シフトメモリーコマンドの選択画面を示した図である。

【図９】

コマンドジェネレーターの、テーブル処理コマンドの製作画面を示した図である。

【図１０】

コマンドジェネレーターの、ＡＮＤ／ＯＲコマンドの製作画面を示した図である。

【図１１】

コマンドジェネレーターの、ラダー図の製作画面を示した図である。

【図１２】

コマンドジェネレーターの、フリッカーコマンドの製作画面を示した図である。

【図１３】

コマンドジェネレーターの、レジスター操作コマンドの選択画面を示した図である。

【図１４】

コマンドジェネレーターの、Ｉレジスター操作コマンドの選択画面を示した図である。

【図１５】

コマンドジェネレーターの、Ｃレジスター操作コマンドの選択画面を示した図である。

【図１６】

コマンドジェネレーターの、その他の関数コマンドの選択画面を示した図である。

【図１７】

コマンドジェネレーターの、通信、アナログコマンドの選択画面を示した図である。

【図１８】

コマンドジェネレーターの、タイマーコマンドの選択画面を示した図である。

【図１９】

コマンドジェネレーターの、タスクコマンドの選択画面を示した図である。

【図２０】

コマンドジェネレーターの、キー入力、表示コマンドの選択画面を示した図である。

【図２１】

コマンドジェネレーターの、ジャンプコマンドの選択画面を示した図である。

【図２２】

コマンドジェネレーターの、信号入出力コマンドの選択画面を示した図である。

【図２３】

簡易コンパイラーの、タスク命令一覧を示した図である。

【図２４】

簡易コンパイラーの、ジャンプ命令一覧を示した図である。

【図２５】

簡易コンパイラーの、Ｉ／Ｏ命令一覧を示した図である。

【図２６】

簡易コンパイラーの、タイマー命令一覧を示した図である。

【図２７】

簡易コンパイラーの、Ａ／Ｄ、Ｄ／Ａ、送信、受信命令一覧を示した図である。

【図２８】

簡易コンパイラーの、演算命令一覧を示した図である。

【図２９】

簡易コンパイラーの、キー入力、ＬＣＤ表示命令一覧を示した図である。

【図３０】

簡易コンパイラーの、その他の命令一覧を示した図である。

【図３１】

コントローラーの、キー配置とそのアスキーコード対応を示した図である。

【図３２】

コントローラーの、サービスメニュー画面を示した図である。

【図３３】

コントローラーの、キーとその役目、表示画面を示した図である。

（Ｆ１，Ｆ２キーの意味はユーザーが決める）

【図３４】

コントローラーの、コントロールメニュー画面を示した図である。

【図３５】

コントロール部の、ＰＣよりオブジェクトプログラムを受信する画面を示した図である。

【図３６】

コントロール部の、受信したオブジェクトプログラムをフラッシュメモリーへ書き込む画面を示した図である。

【図３７】

コントローラーの、設定メニュー画面を示した図である。

【図３８】

設定部の、データメニューにデータをセットする画面を示した図である。

【図３９】

設定部の、オーバータイマーにタイマー値をセットする画面を示した図である。

【図４０】

設定部の、通信０ＣＨのパラメーターをセットする画面を示した図である。

【図４１】

設定部の、通信１ＣＨのパラメーターをセットする画面を示した図である。

【図４２】

設定部の、通信０ＣＨ，１ＣＨのターミネーターと受信オーバータイムを設定する画面を示した図である。

【図４３】

設定部の、カレンダークロックを設定する画面を示した図である。

【図４４】

設定部の、Ａ／Ｄ、Ｄ／Ａ、オプション個数を設定する画面を示した図である。

【図４５】

コントローラーの、チェックメニュー画面を示した図である。

【図４６】

チェック部の、Ｉ／Ｏチェック画面を示した図である。

【図４７】

チェック部の、Ａ／Ｄチェック画面を示した図である。

【図４８】

チェック部の、Ｄ／Ａチェック画面を示した図である。

【図４９】

チェック部の、受信データ、０ＣＨ，１ＣＨ画面を示した図である。

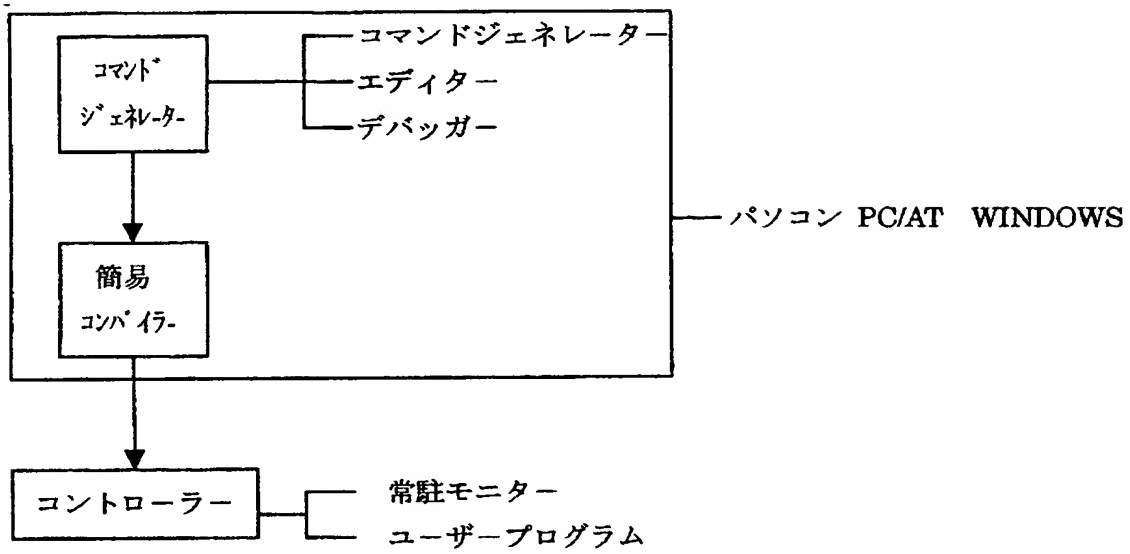
【図５０】

チェック部の、その他のチェック画面を示した図である。

（ＭＡＸサイクルタイム、Ｉ，Ｃ，Ｒレジスター内容、内部信号Ｖの内容）

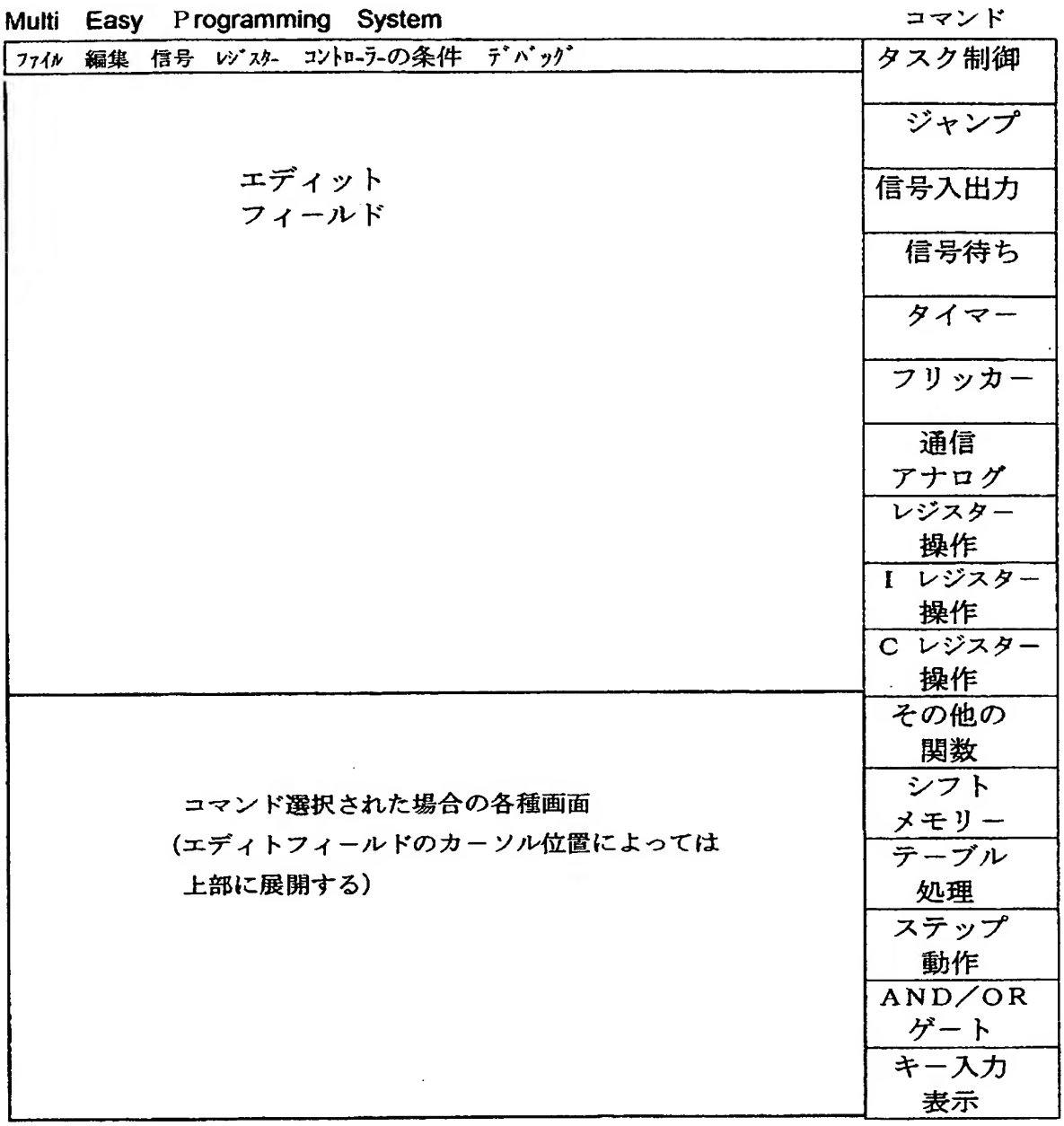
【書類名】図面

【図１】



コントローラー仕様：縦１７０ｍｍ　ｘ　横１２０ｍｍ　ｘ　高さ６０ｍｍ
キー12入力、LCD 6×16桁、I/O各32点フォトカプラー付き、A/D 8CH、D/A 2CH、カレンダー時計、RS232C、RS422、ＣＰＵ　Ｚ８０　１０　ｍＨＺ、フラッシュメモリー５１２ＫＢ、スタテイックＲＡＭ　５１２ＫＢ　バッテリーバックアップ付き

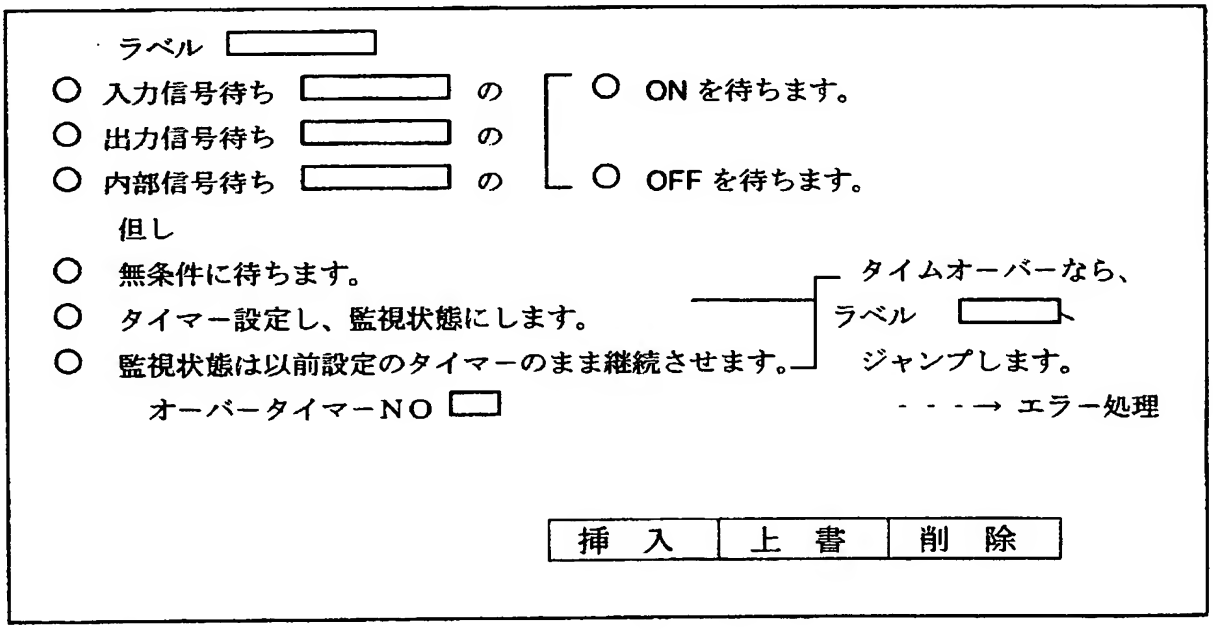
【図２】


Multi Easy Programming System
コマンド
ファイル 編集 信号 レジスター コントローラーの条件 デバッグ
エディット
フィールド
コマンド選択された場合の各種画面
(エディトフィールドのカーソル位置によっては
上部に展開する)
タスク制御
ジャンプ
信号入出力
信号待ち
タイマー
フリッカー
通信
アナログ
レジスター
操作
I レジスター
操作
C レジスター
操作
その他の
関数
シフト
メモリー
テーブル
処理
ステップ
動作
AND／OR
ゲート
キー入力
表示

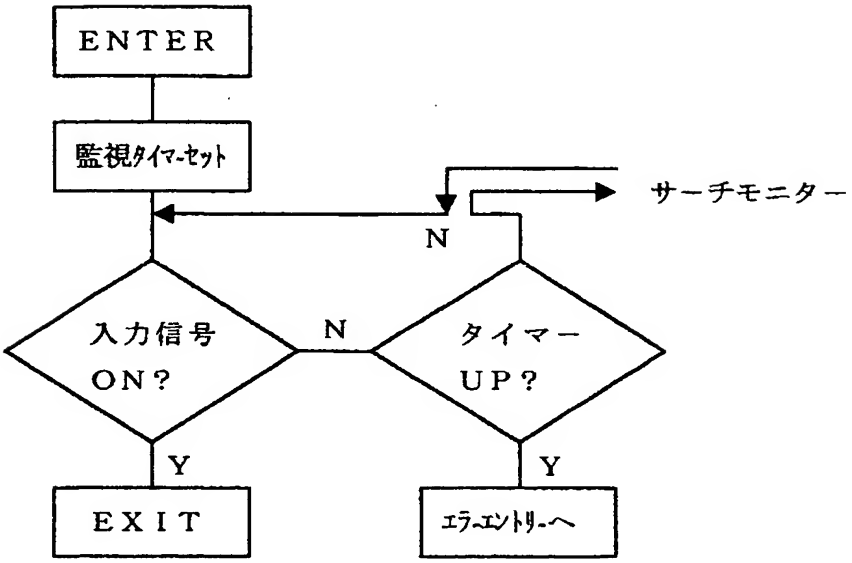
【図３】

番号	設定	入力信号名	注釈	番号	設定	入力信号名	注釈
0				１６			
1				１７			
2				１８			
3				１９			
4				２０			
5				２１			
6				２２			
7				２３			
8				２４			
9				２５			
１０				２６			
１１				２７			
１２				２８			
１３				２９			
１４				３０			
１５				３１			

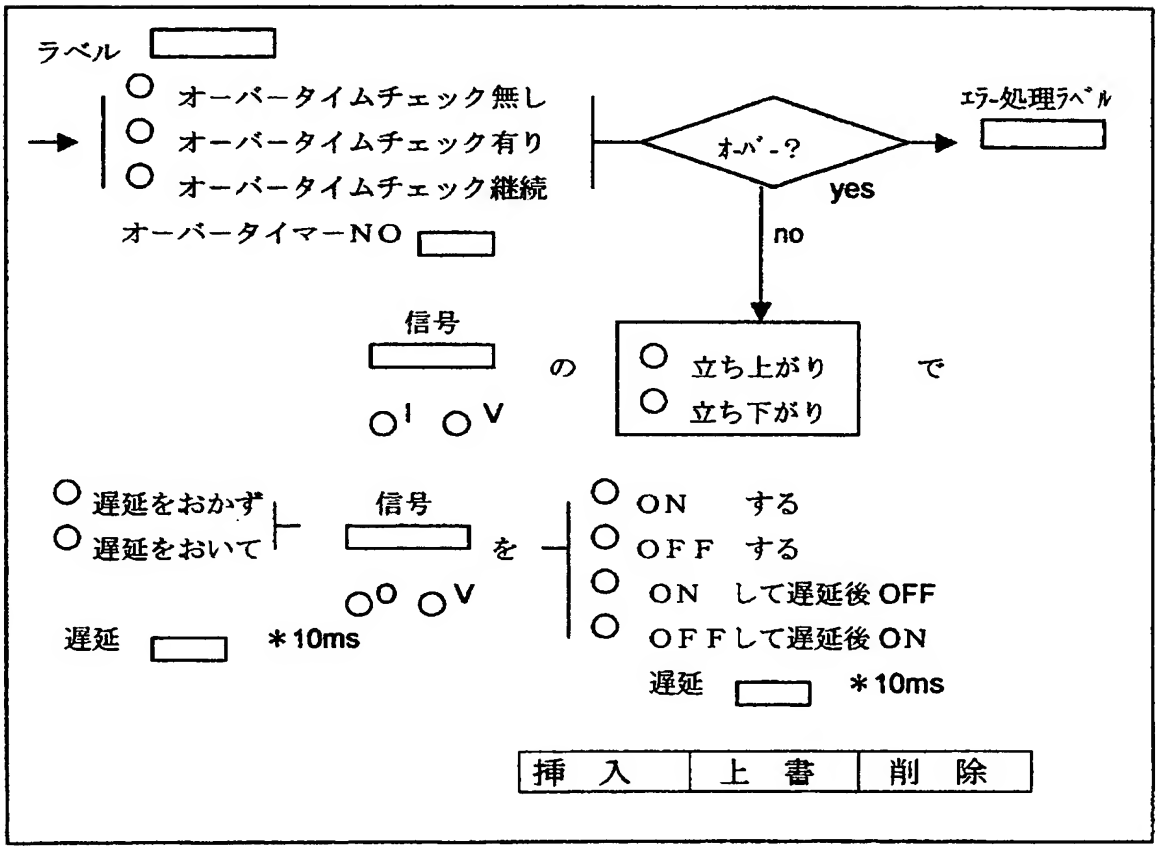
【図４】



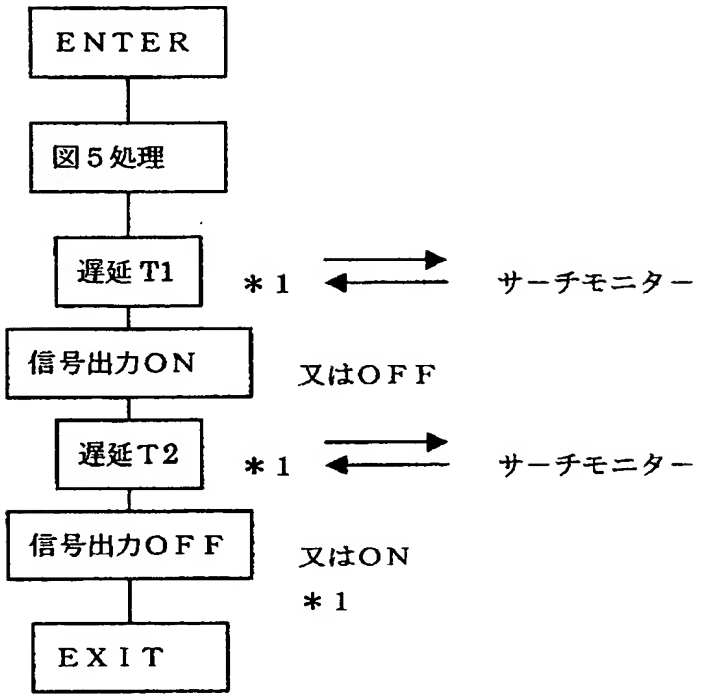
【図５】


ENTER
監視ﾀｲﾏｰｾｯﾄ
サーチモニター
N
入力信号
ON？
N
タイマー
UP？
Y
Y
EXIT
ｴﾗｰｴﾝﾄﾘｰへ

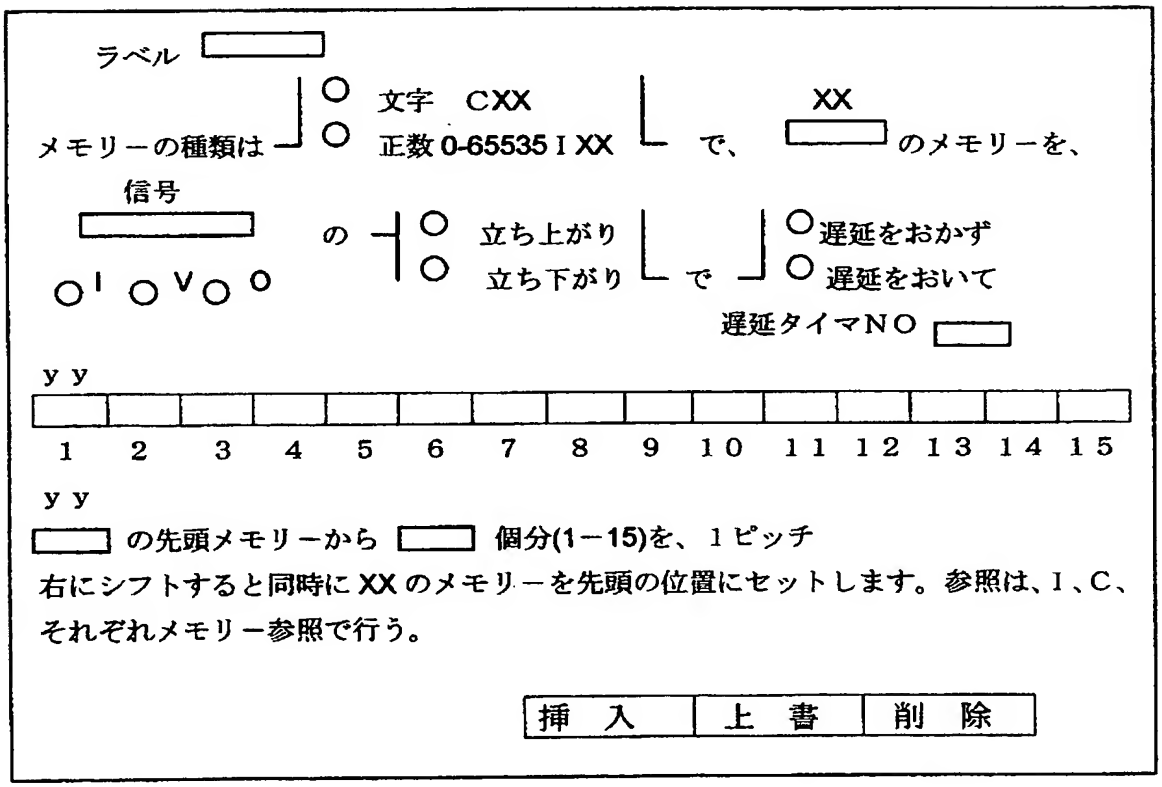
【図６】


ラベル
オーバータイムチェック無し
オーバータイムチェック有り
オーバータイムチェック継続
オーバータイマーNO
ｵｰﾊﾞｰ？
yes
no
ｴﾗｰ処理ﾗﾍﾞﾙ
信号
の
立ち上がり
立ち下がり
で
遅延をおかず
遅延をおいて
信号
を
ON　する
OFF　する
ON　して遅延後 OFF
OFFして遅延後 ON
遅延
＊10ms
遅延
＊10ms
挿　入
上　書
削　除

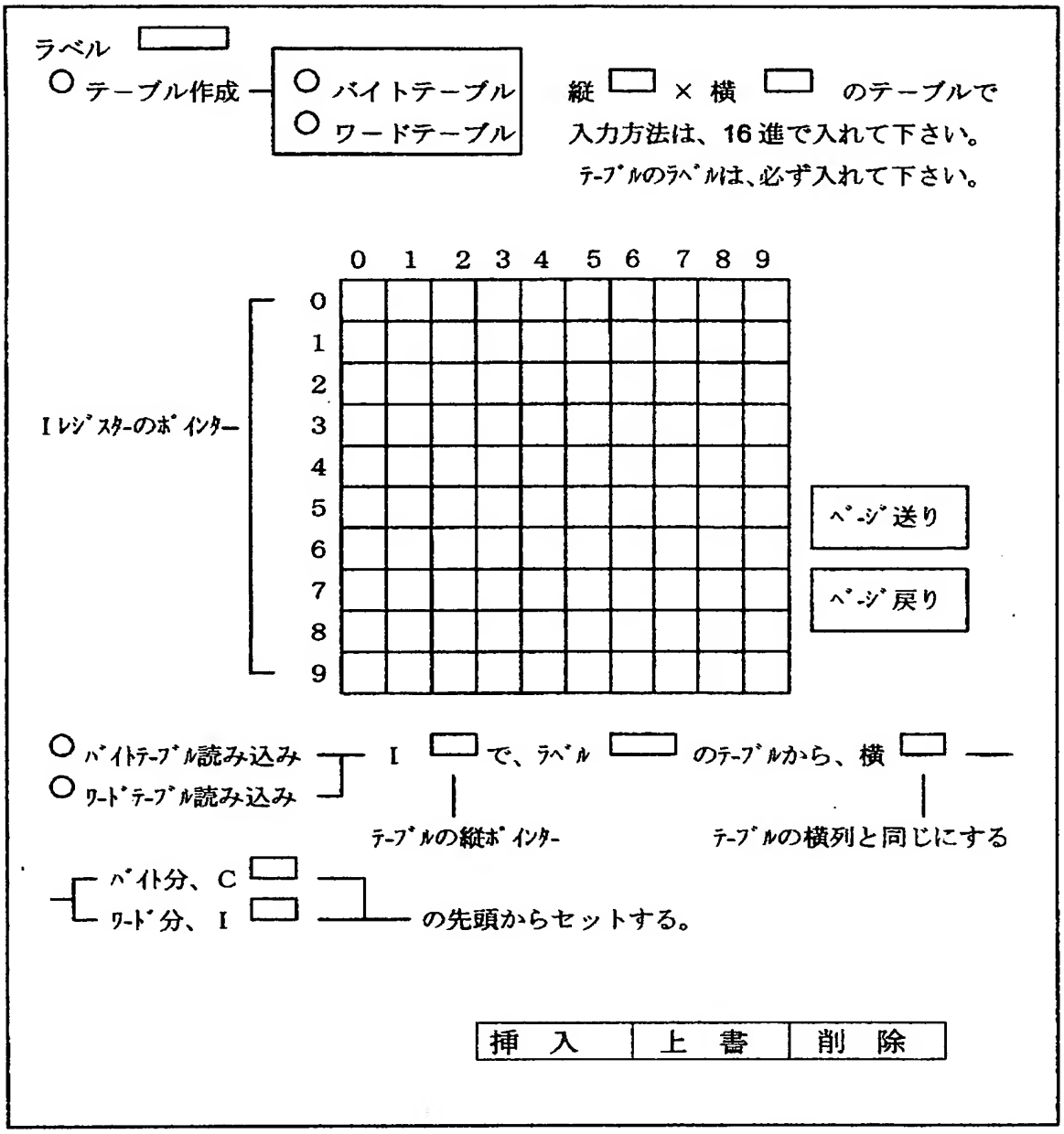
【図７】


ENTER
図５処理
遅延T1
＊１
サーチモニター
信号出力ON
又はＯＦＦ
遅延T2
＊１
サーチモニター
信号出力ＯＦＦ
又はON
＊１
EXIT

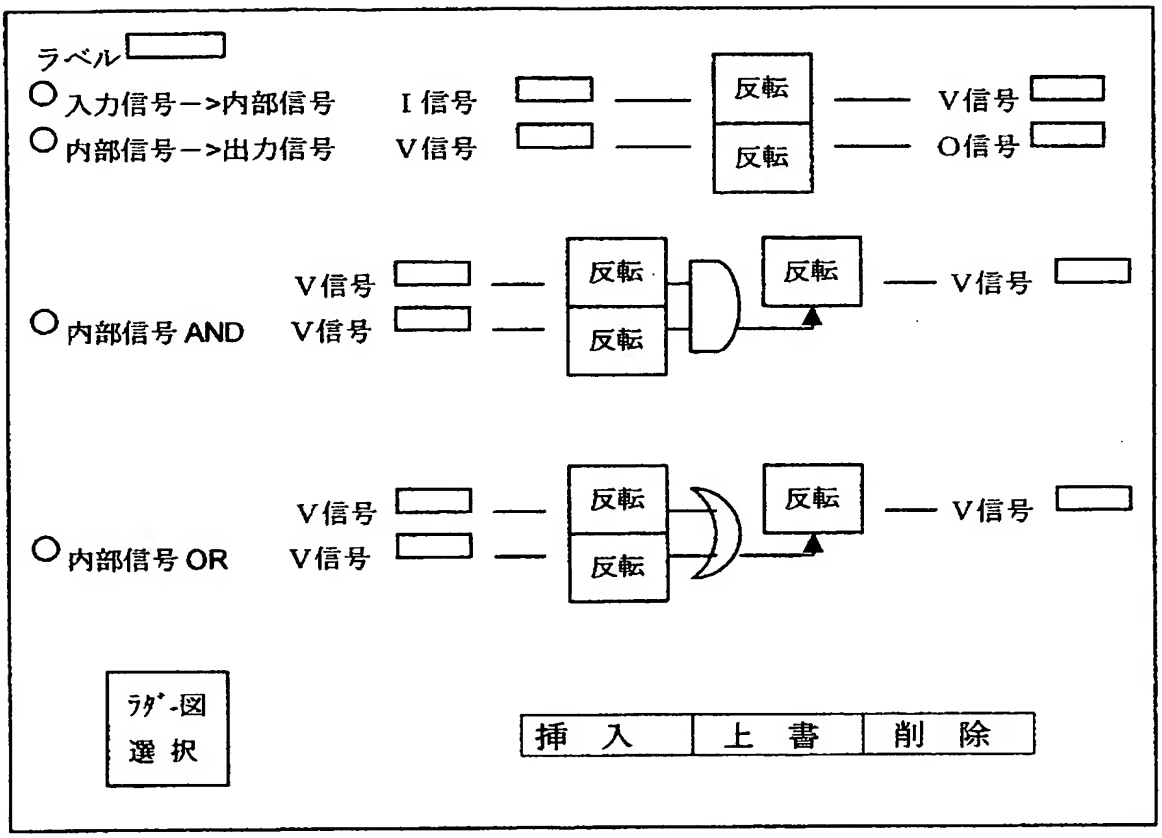
【図８】


ラベル
メモリーの種類は
文字　CXX
正数 0-65535 I XX
で、
XX
のメモリーを、
信号
の
立ち上がり
立ち下がり
で
遅延をおかず
遅延をおいて
遅延タイマNO
y y
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
y y
の先頭メモリーから
個分(1－15)を、１ピッチ
右にシフトすると同時に XX のメモリーを先頭の位置にセットします。参照は、I、C、
それぞれメモリー参照で行う。
挿　入
上　書
削　除

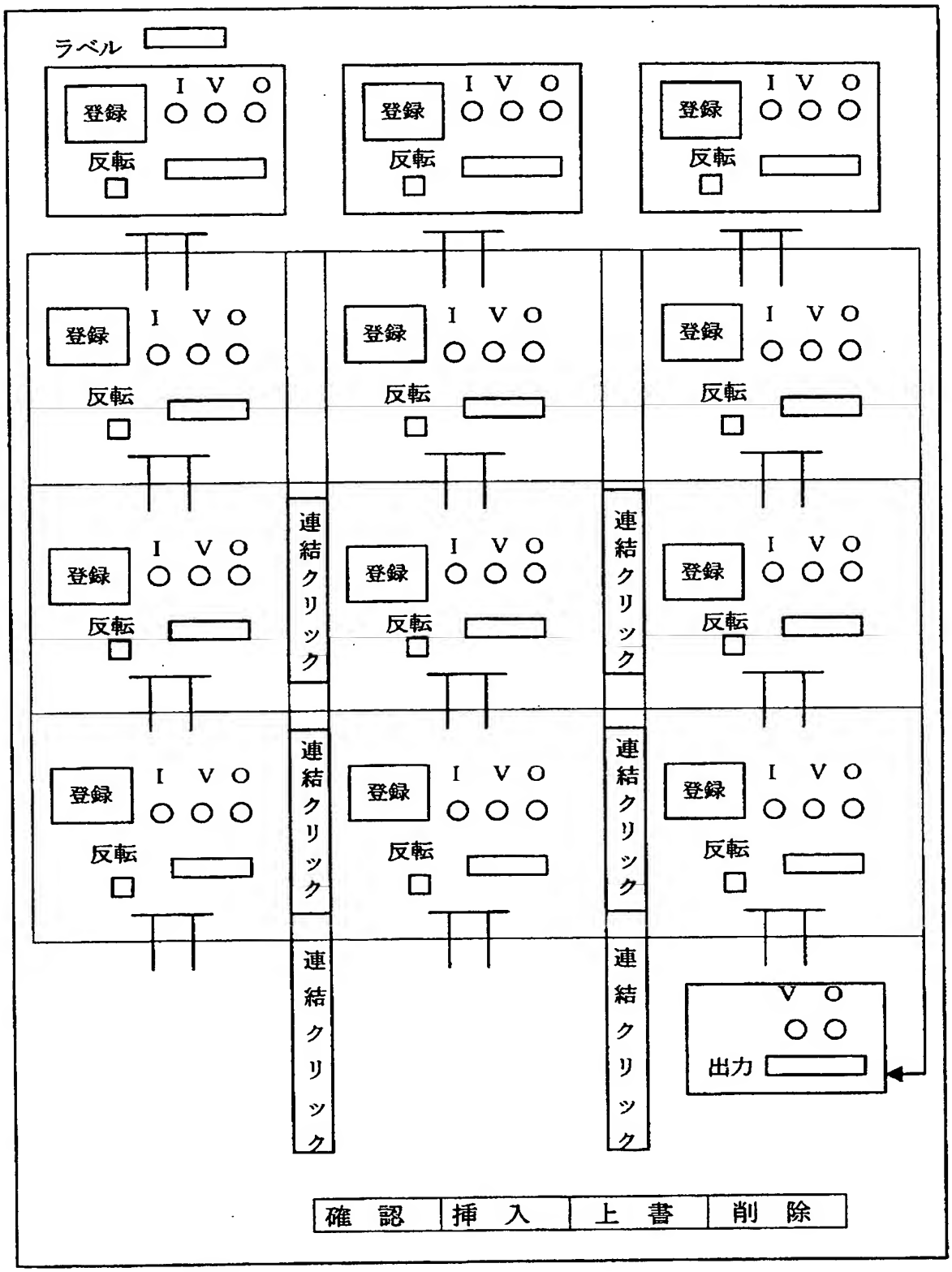
【図９】


ラベル
テーブル作成
バイトテーブル
ワードテーブル
縦 × 横 のテーブルで
入力方法は、16進で入れて下さい。
ﾃｰﾌﾞﾙのﾗﾍﾞﾙは、必ず入れて下さい。
0 1 2 3 4 5 6 7 8 9
Iﾚｼﾞｽﾀｰのﾎﾟｲﾝﾀｰ
ﾍﾟｰｼﾞ送り
ﾍﾟｰｼﾞ戻り
ﾊﾞｲﾄﾃｰﾌﾞﾙ読み込み
ﾜｰﾄﾞﾃｰﾌﾞﾙ読み込み
I で、ﾗﾍﾞﾙ のﾃｰﾌﾞﾙから、横
ﾃｰﾌﾞﾙの縦ﾎﾟｲﾝﾀｰ
ﾃｰﾌﾞﾙの横列と同じにする
ﾊﾞｲﾄ分、C
ﾜｰﾄﾞ分、I
の先頭からセットする。
挿　入
上　書
削　除

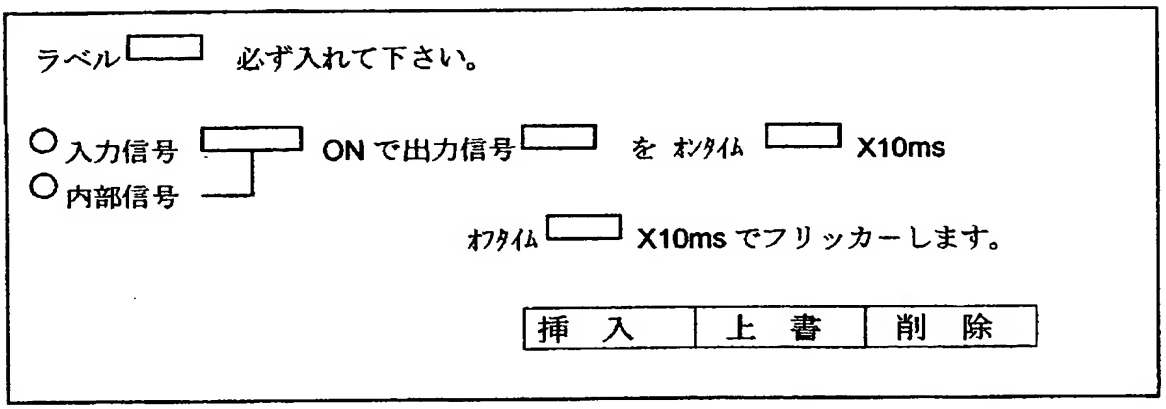
【図１０】


ラベル
入力信号－>内部信号
内部信号－>出力信号
I信号
V信号
反転
反転
V信号
O信号
内部信号 AND
V信号
V信号
反転
反転
反転
V信号
内部信号 OR
V信号
V信号
反転
反転
反転
V信号
ﾗﾀﾞｰ図
選 択
挿 入
上 書
削 除

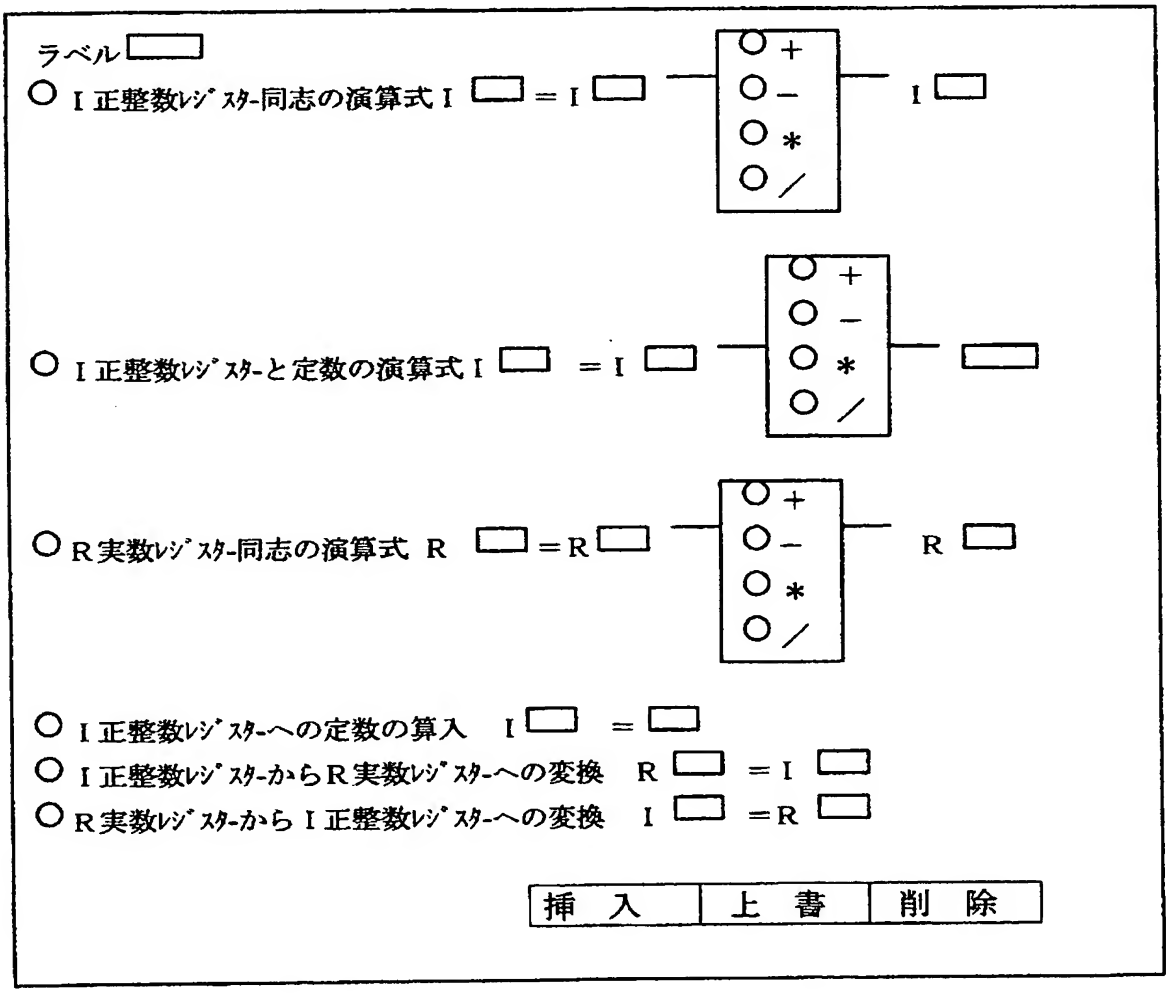
【図１１】



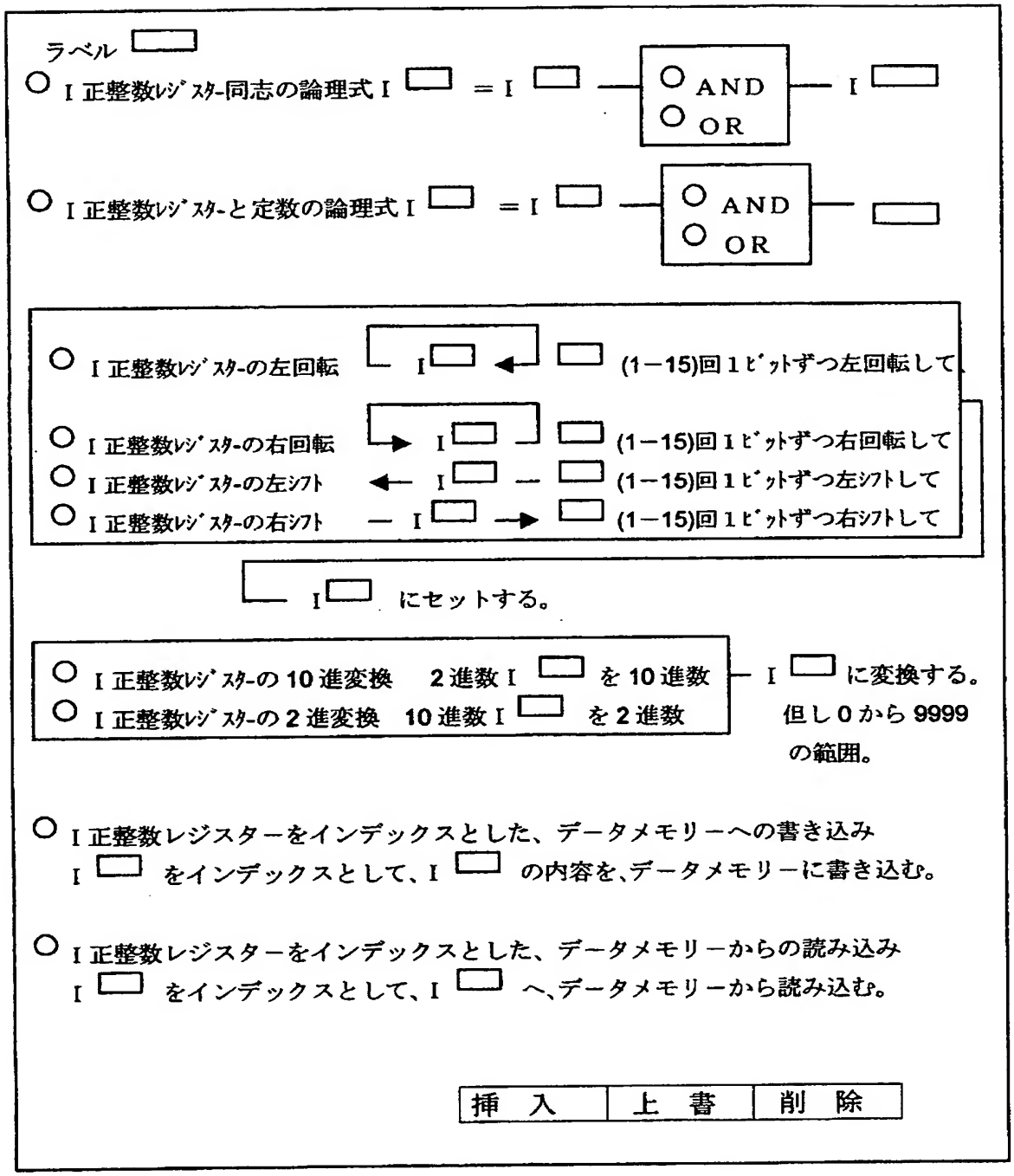
【図１２】


ラベル
必ず入れて下さい。
入力信号
内部信号
ON で出力信号
を ｵﾝﾀｲﾑ
X10ms
ｵﾌﾀｲﾑ
X10ms でフリッカーします。
挿　入
上　書
削　除

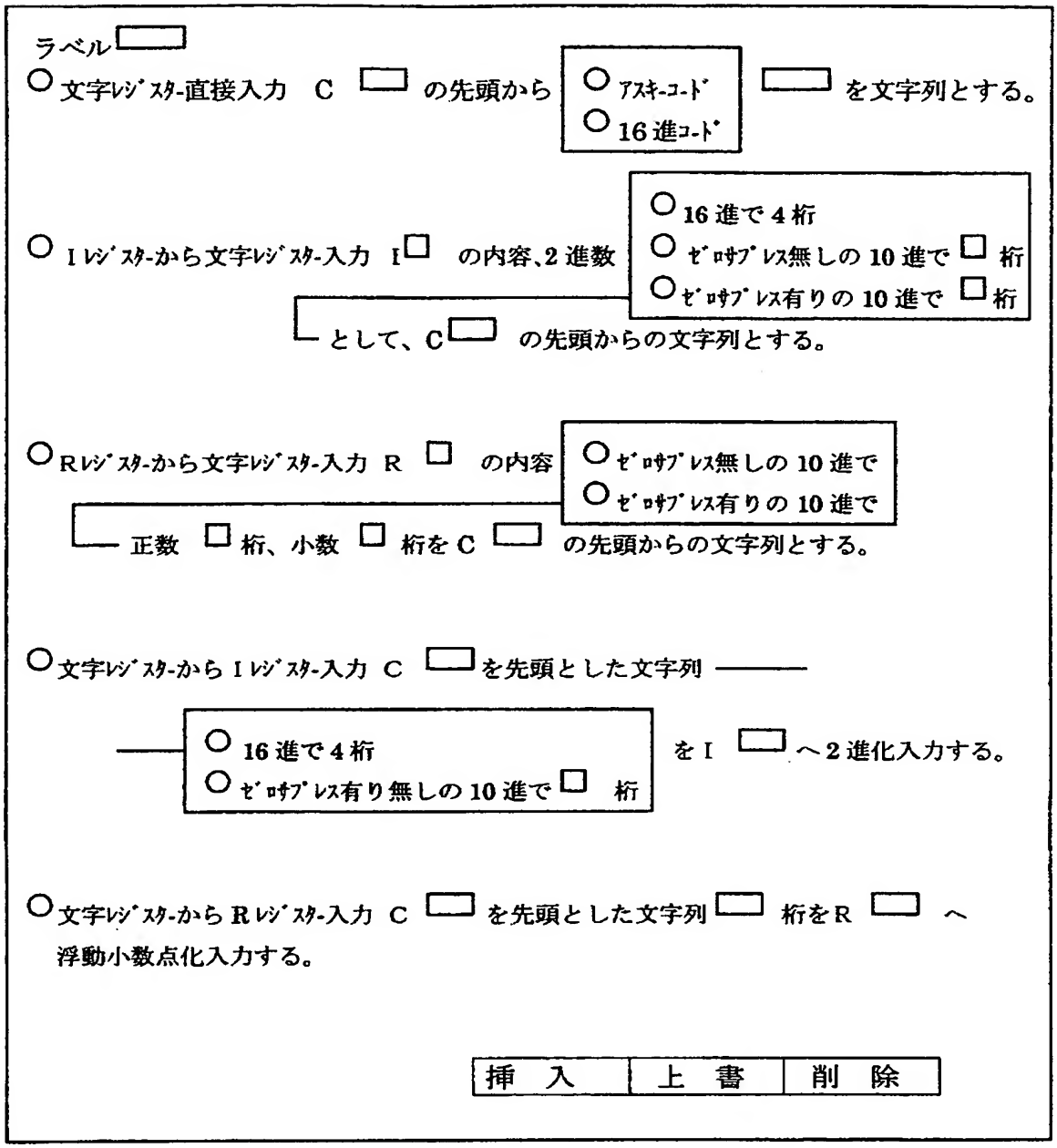
【図１３】


ラベル
Ｉ正整数ﾚｼﾞｽﾀｰ同志の演算式 I = I
I
Ｉ正整数ﾚｼﾞｽﾀｰと定数の演算式 I = I
Ｒ実数ﾚｼﾞｽﾀｰ同志の演算式 R = R
R
Ｉ正整数ﾚｼﾞｽﾀｰへの定数の算入 I =
Ｉ正整数ﾚｼﾞｽﾀｰからＲ実数ﾚｼﾞｽﾀｰへの変換 R = I
Ｒ実数ﾚｼﾞｽﾀｰからＩ正整数ﾚｼﾞｽﾀｰへの変換 I = R
挿　入
上　書
削　除

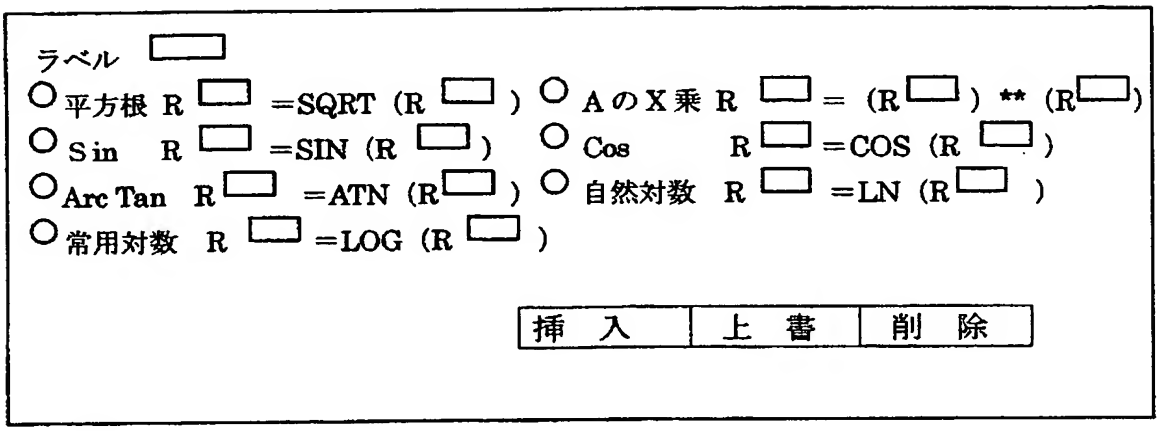
【図１４】


ラベル
○ I正整数ﾚｼﾞｽﾀ-同志の論理式 I = I
○ AND
○ OR
I
○ I正整数ﾚｼﾞｽﾀ-と定数の論理式 I = I
○ AND
○ OR
○ I正整数ﾚｼﾞｽﾀ-の左回転 I (1−15)回１ﾋﾞｯﾄずつ左回転して
○ I正整数ﾚｼﾞｽﾀ-の右回転 I (1−15)回１ﾋﾞｯﾄずつ右回転して
○ I正整数ﾚｼﾞｽﾀ-の左ｼﾌﾄ I (1−15)回１ﾋﾞｯﾄずつ左ｼﾌﾄして
○ I正整数ﾚｼﾞｽﾀ-の右ｼﾌﾄ I (1−15)回１ﾋﾞｯﾄずつ右ｼﾌﾄして
I にセットする。
○ I正整数ﾚｼﾞｽﾀ-の10進変換　2進数 I を10進数
○ I正整数ﾚｼﾞｽﾀ-の2進変換　10進数 I を2進数
I に変換する。
但し0から9999の範囲。
○ I正整数レジスターをインデックスとした、データメモリーへの書き込み
I をインデックスとして、I の内容を、データメモリーに書き込む。
○ I正整数レジスターをインデックスとした、データメモリーからの読み込み
I をインデックスとして、I へ、データメモリーから読み込む。
挿　入
上　書
削　除

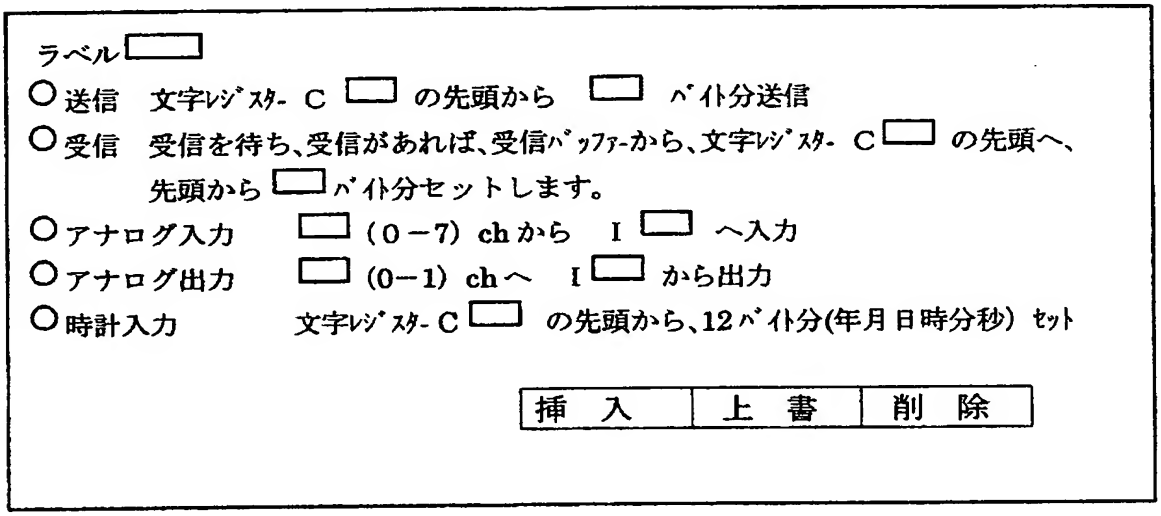
【図１５】


ラベル
○文字ﾚｼﾞｽﾀ-直接入力　C　の先頭から
○ｱｽｷｰｺｰﾄﾞ
○16進ｺｰﾄﾞ
を文字列とする。
○Iﾚｼﾞｽﾀ-から文字ﾚｼﾞｽﾀ-入力　I　の内容､2進数
○16進で4桁
○ｾﾞﾛｻﾌﾟﾚｽ無しの10進で　桁
○ｾﾞﾛｻﾌﾟﾚｽ有りの10進で　桁
として、C　の先頭からの文字列とする。
○Rﾚｼﾞｽﾀ-から文字ﾚｼﾞｽﾀ-入力　R　の内容
○ｾﾞﾛｻﾌﾟﾚｽ無しの10進で
○ｾﾞﾛｻﾌﾟﾚｽ有りの10進で
正数　桁、小数　桁をC　の先頭からの文字列とする。
○文字ﾚｼﾞｽﾀ-からIﾚｼﾞｽﾀ-入力　C　を先頭とした文字列
○16進で4桁
○ｾﾞﾛｻﾌﾟﾚｽ有り無しの10進で　桁
をI　へ2進化入力する。
○文字ﾚｼﾞｽﾀ-からRﾚｼﾞｽﾀ-入力　C　を先頭とした文字列　桁をR　へ
浮動小数点化入力する。
挿　入
上　書
削　除

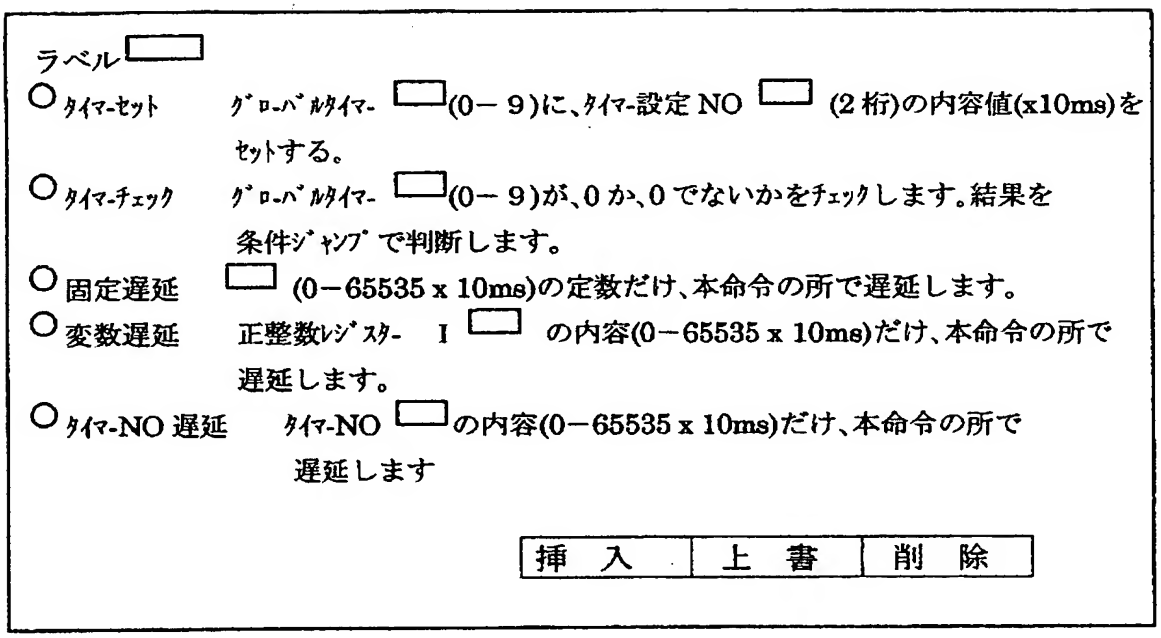
【図１６】


ラベル
○平方根 R □ =SQRT (R □)
○AのX乗 R □ = (R□) ** (R□)
○Sin R □ =SIN (R □)
○Cos R □ =COS (R □)
○Arc Tan R□ =ATN (R□)
○自然対数 R □ =LN (R□)
○常用対数 R □ =LOG (R □)
挿 入
上 書
削 除

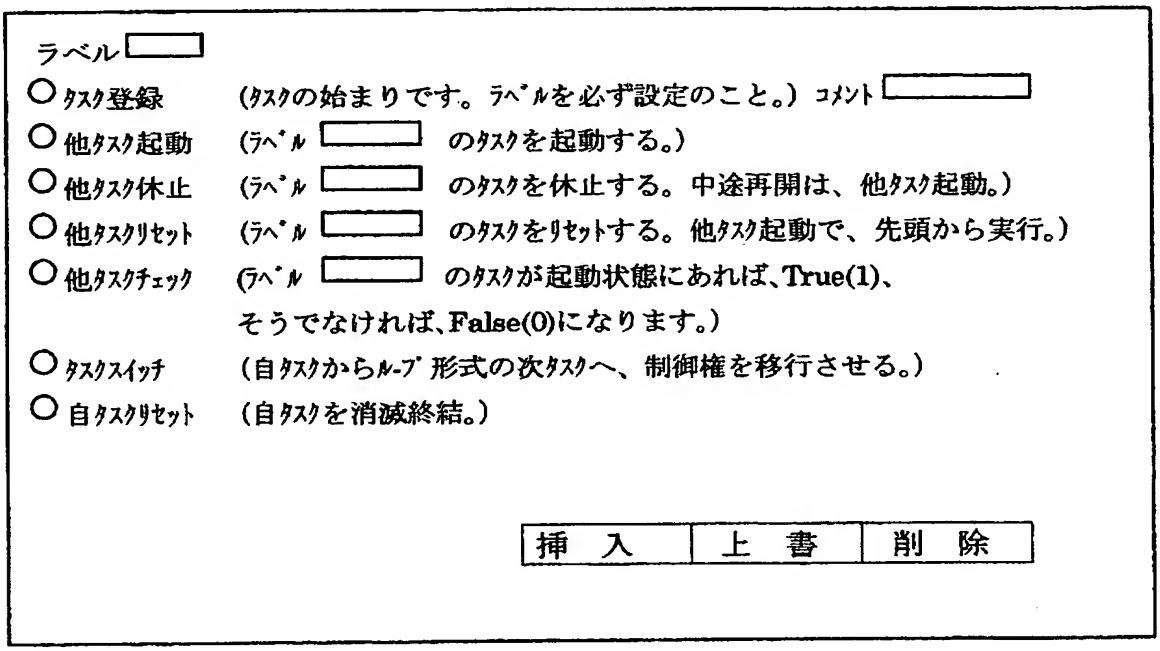
【図１７】


ラベル□
○送信 文字ﾚｼﾞｽﾀ- C □ の先頭から □ ﾊﾞｲﾄ分送信
○受信 受信を待ち、受信があれば、受信ﾊﾞｯﾌｧ-から、文字ﾚｼﾞｽﾀ- C□ の先頭へ、
先頭から□ﾊﾞｲﾄ分セットします。
○アナログ入力 □ (0－7) chから I □ へ入力
○アナログ出力 □ (0－1) chへ I□ から出力
○時計入力 文字ﾚｼﾞｽﾀ- C□ の先頭から、12ﾊﾞｲﾄ分(年月日時分秒) ｾｯﾄ
挿 入
上 書
削 除

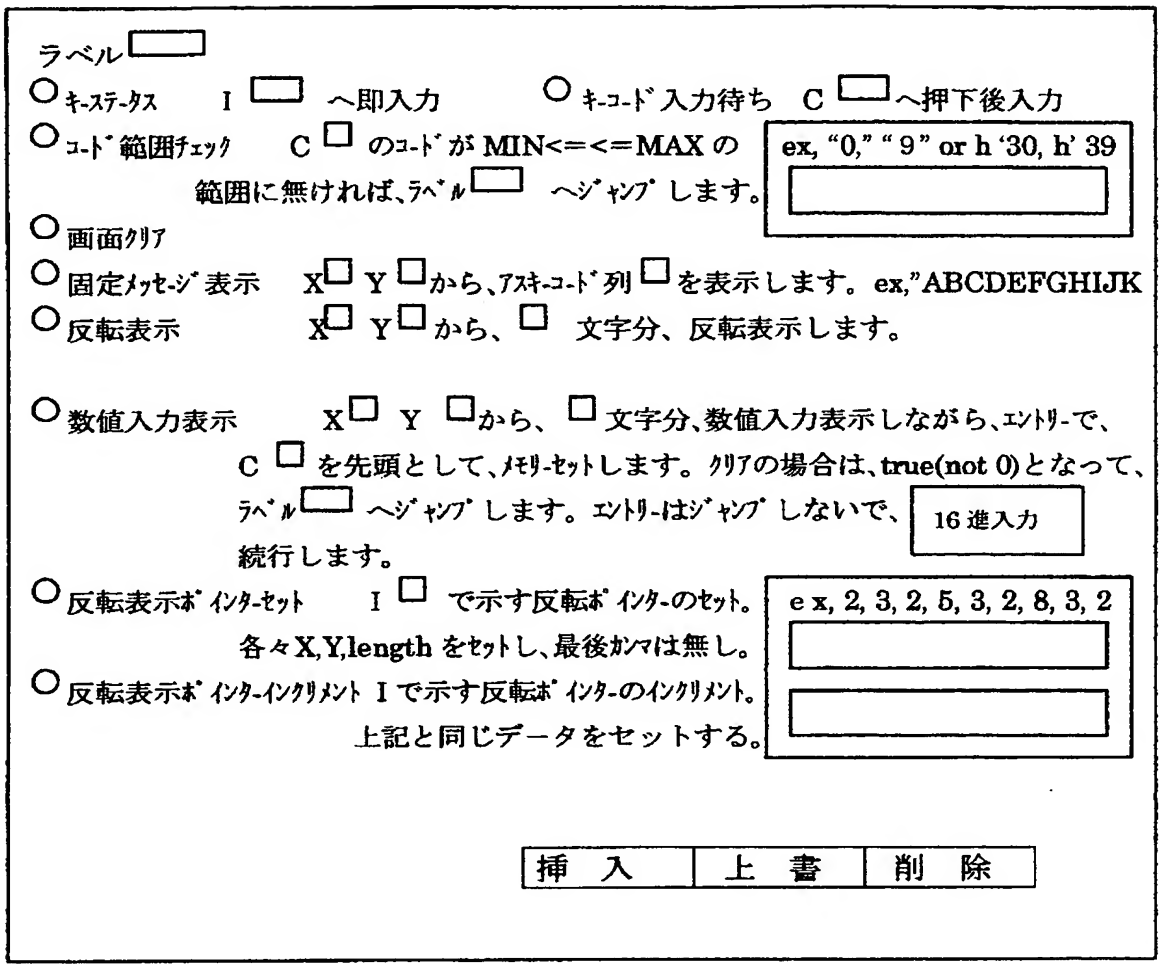
【図１８】


ラベル
○ﾀｲﾏｰｾｯﾄ　ｸﾞﾛｰﾊﾞﾙﾀｲﾏｰ (0－9)に、ﾀｲﾏｰ設定 NO (2 桁)の内容値(x10ms)をｾｯﾄする。
○ﾀｲﾏｰﾁｪｯｸ　ｸﾞﾛｰﾊﾞﾙﾀｲﾏｰ (0－9)が、0 か、0 でないかをﾁｪｯｸします。結果を条件ｼﾞｬﾝﾌﾟで判断します。
○固定遅延　(0－65535 x 10ms)の定数だけ、本命令の所で遅延します。
○変数遅延　正整数ﾚｼﾞｽﾀｰ I の内容(0－65535 x 10ms)だけ、本命令の所で遅延します。
○ﾀｲﾏｰNO 遅延　ﾀｲﾏｰNO の内容(0－65535 x 10ms)だけ、本命令の所で遅延します
挿　入
上　書
削　除

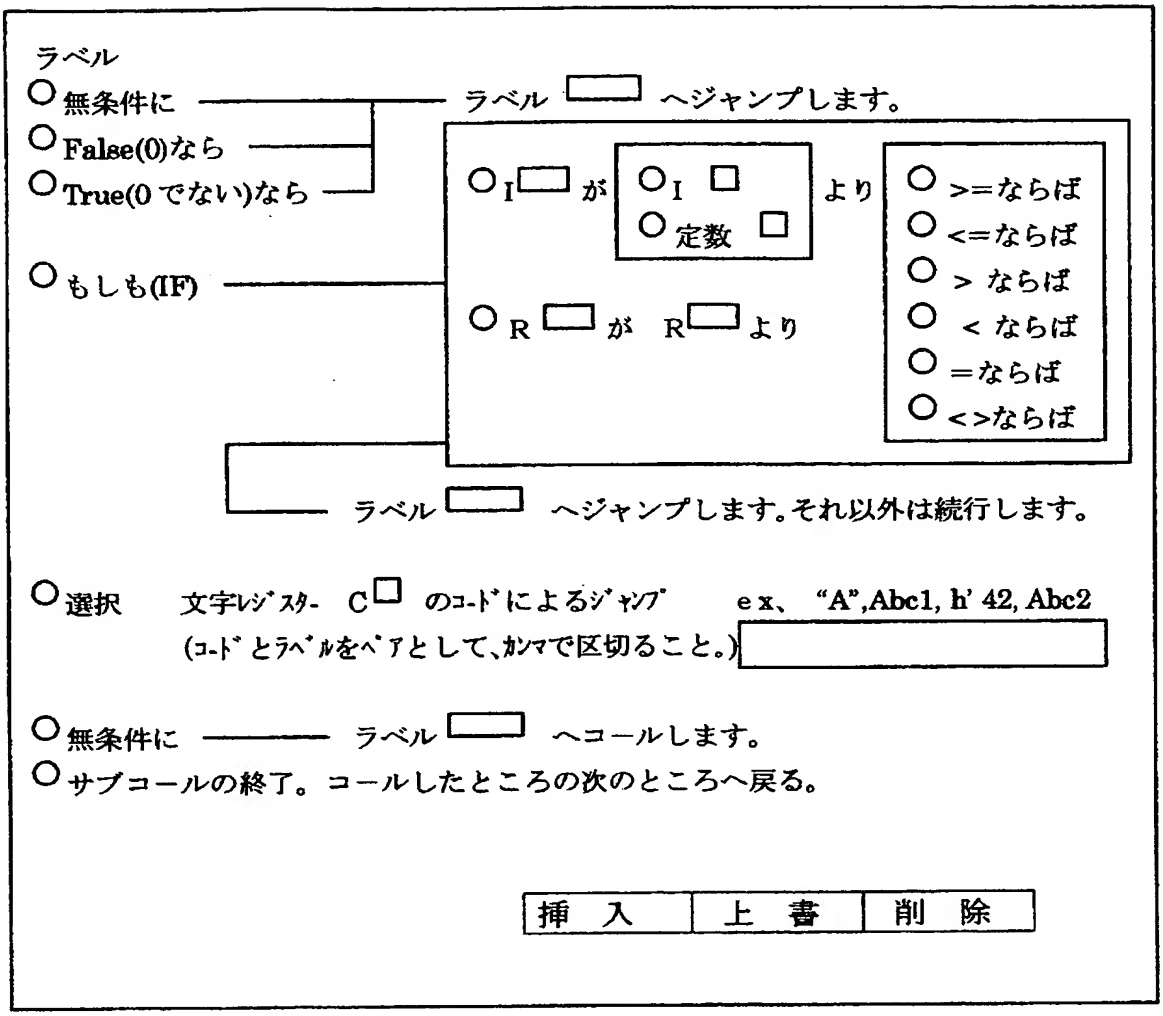
【図１９】


ラベル
○ﾀｽｸ登録　(ﾀｽｸの始まりです。ﾗﾍﾞﾙを必ず設定のこと。) ｺﾒﾝﾄ
○他ﾀｽｸ起動　(ﾗﾍﾞﾙ のﾀｽｸを起動する。)
○他ﾀｽｸ休止　(ﾗﾍﾞﾙ のﾀｽｸを休止する。中途再開は、他ﾀｽｸ起動。)
○他ﾀｽｸﾘｾｯﾄ　(ﾗﾍﾞﾙ のﾀｽｸをﾘｾｯﾄする。他ﾀｽｸ起動で、先頭から実行。)
○他ﾀｽｸﾁｪｯｸ　(ﾗﾍﾞﾙ のﾀｽｸが起動状態にあれば、True(1)、そうでなければ、False(0)になります。)
○ﾀｽｸｽｲｯﾁ　(自ﾀｽｸからﾙｰﾌﾟ形式の次ﾀｽｸへ、制御権を移行させる。)
○自ﾀｽｸﾘｾｯﾄ　(自ﾀｽｸを消滅終結。)
挿　入
上　書
削　除

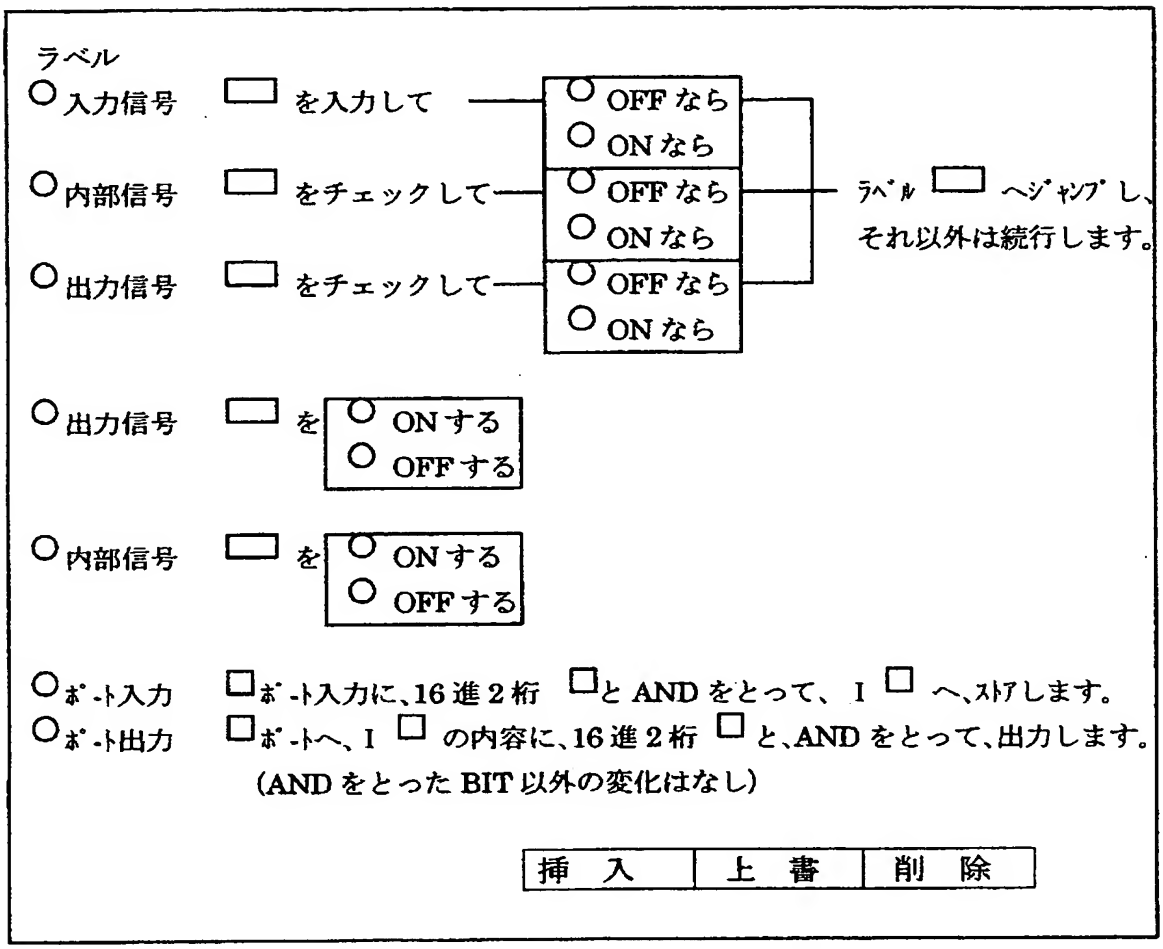
【図２０】


ラベル
○ｷｰｽﾃｰﾀｽ　I　へ即入力
○ｷｰｺｰﾄﾞ入力待ち　C　へ押下後入力
○ｺｰﾄﾞ範囲ﾁｪｯｸ　C　のｺｰﾄﾞがMIN<=<=MAXの範囲に無ければ、ﾗﾍﾞﾙ　へｼﾞｬﾝﾌﾟします。
ex, “0,” “9” or h ‘30, h’ 39
○画面ｸﾘｱ
○固定ﾒｯｾｰｼﾞ表示　X　Y　から、ｱｽｷｰｺｰﾄﾞ列　を表示します。ex,”ABCDEFGHIJK
○反転表示　X　Y　から、　文字分、反転表示します。
○数値入力表示　X　Y　から、　文字分、数値入力表示しながら、ｴﾝﾄﾘｰで、C　を先頭として、ﾒﾓﾘｰｾｯﾄします。ｸﾘｱの場合は、true(not 0)となって、ﾗﾍﾞﾙ　へｼﾞｬﾝﾌﾟします。ｴﾝﾄﾘｰはｼﾞｬﾝﾌﾟしないで、続行します。
16進入力
○反転表示ﾎﾟｲﾝﾀｰｾｯﾄ　I　で示す反転ﾎﾟｲﾝﾀｰのｾｯﾄ。各々X,Y,lengthをｾｯﾄし、最後ｶﾝﾏは無し。
ex, 2, 3, 2, 5, 3, 2, 8, 3, 2
○反転表示ﾎﾟｲﾝﾀｰｲﾝｸﾘﾒﾝﾄ　I　で示す反転ﾎﾟｲﾝﾀｰのｲﾝｸﾘﾒﾝﾄ。上記と同じデータをｾｯﾄする。
挿　入
上　書
削　除

【図２１】


ラベル
○無条件に
○False(0)なら
○True(0でない)なら
ラベル □ へジャンプします。
○もしも(IF)
○I□ が ○I □ ○定数 □ より
○>=ならば
○<=ならば
○ > ならば
○ < ならば
○ =ならば
○ <>ならば
○R□ が R□ より
ラベル □ へジャンプします。それ以外は続行します。
○選択 文字ﾚｼﾞｽﾀ- C□ のｺｰﾄﾞによるｼﾞｬﾝﾌﾟ ex、 "A",Abc1, h' 42, Abc2
(ｺｰﾄﾞとﾗﾍﾞﾙをﾍﾟｱとして､ｶﾝﾏで区切ること。)
○無条件に ラベル □ へコールします。
○サブコールの終了。コールしたところの次のところへ戻る。
挿　入
上　書
削　除

【図２２】



【図２３】

T	タスクの登録と開始
Act_Taskn	別タスクの起動
Pause_Taskn	別タスクの休止
Reset_Taskn	別タスクのリセット
Stat?_Taskn	別タスクが起動しているかチェック
S	自タスクから、次タスクへスイッチする
X	自タスクの自己終了と消滅

【図２４】

Ｇｏ＿Ｐｃｌａｂｅｌ	無条件ジャンプ	
Ｇｔ＿Ｐｃｌａｂｅｌ	条件ジャンプ、　０でない時ジャンプ	
Ｇｆ＿Ｐｃｌａｂｅｌ	条件ジャンプ、　０の時ジャンプ	
Ｃａｌｌ＿Ｐｃｌａｂｅｌ	サブルーチン　　コール	
Ｒ	サブルーチン　　リターン	
Ｓｅｌ＿ｃ10, "1, Ｐｃ1, … " 2, Ｐｃ2…	ｃ10が "1" ならＰｃ1、"2" ならＰｃ2へジャンプ。それ以外は続行	
Ｉｆ＿ｉ10＝ｉ20,Pclabel Ｉｆ＿ｉ10<＞ｉ20,Pclabel Ｉｆ＿ｉ10＞＝ｉ20,Pclabel Ｉｆ＿ｉ10<＝ｉ20,Pclabel Ｉｆ＿ｉ10＞ｉ20,Pclabel Ｉｆ＿ｉ10<ｉ20,Pclabel Ｉｆ＿ｉ10＝999,Pclabel Ｉｆ＿ｉ10<＞999,Pclabel Ｉｆ＿ｉ10＞＝999,Pclabel Ｉｆ＿ｉ10<＝999,Pclabel	Ｉｆ＿ｉ10＞999,Pclabel Ｉｆ＿ｉ10<9999,Pclabel Ｉｆ＿ｒ10＝ｒ20,Pclabel Ｉｆ＿ｒ10<＞ｒ20,Pclabel Ｉｆ＿ｒ10＞＝ｒ20,Pclabel Ｉｆ＿ｒ10<＝ｒ20,Pclabel Ｉｆ＿ｒ10＞ｒ20,Pclabel Ｉｆ＿ｒ10<ｒ20,Pclabel	Ｉｆ(もしも)の条件に合えばＰｃｌａｂｅｌへジャンプ

【図２５】

Ｉ？＿　Ｉｓｉｇ	Ｉsig入力のON／OFFをチェック	
Ｏ？＿　Ｏｓｉｇ	Osig出力のON／OFFをチェック	
Ｖ？＿　Ｖｓｉｇ	Vsig内部フラッグのON／OFFをチェック	
Ｏｎ＿　Ｏｓｉｇ	Osig出力をONする	
Ｏｆ＿　Ｏｓｉｇ	Osig出力をOFFする	
Ｖｎ＿　Ｖｓｉｇ	Vsig内部フラッグをONする	
Ｖｆ＿　Ｖｓｉｇ	Vsig内部フラッグをOFFする	
Ｐｉ＿3, i10 & h'0f	入力３ポートと16進0fとANDをとって　ｉ10代入 （ｉ10のＨｉバイトは0になる）	
Ｐｏ＿3, i10 & h'7f	ｉ10のＬ0バイトと16進7fとANDをとって出力３ポートへ出力する（出力３ポートのＢ７は変化しない）	
Ｉｎｗ＿Ｉｓｉｇ	Ｉsig入力のONをウェイトする	
Ｉｎｗ＿99,Ｐｃｌａｂｅｌ，Ｉｓｉｇ	Ｉsig入力のONをタイマーNO 99の値の分だけウェイトし、タイムオーバーでPclabelへジャンプし、時間内にONすれば命令続行する	
Ｉｎｗ＿C,Ｐｃｌａｂｅｌ，Ｉｓｉｇ	以前の監視タイマーの続行でＩsig入力のONをウェイト	
Ｉfw＿Ｉsig Ｉfw＿99, Pclabel, Ｉsig Ｉfw＿C, Pclabel, Ｉsig	Ｉnwの反対でOFFをウェイトする	
Ｏnw ＿ Osig Ｏnw ＿ 99, Pclabel,Osig Ｏnw ＿C, Pclabel,　Osig	Ｏfw ＿ Osig Ｏfw ＿ 99, Pclabel,Osig Ｏfw ＿ C, Pclabel,Osig	出力信号のウェイト
Ｖnw ＿ Vsig Ｖnw ＿ 99, Pclabel,Vsig Ｖnw ＿C, Pclabel,Vsig	Ｖfw ＿ Vsig Ｖfw ＿ 99, Pclabel,Vsig Ｖfw ＿ C, Pclabel,Vsig	内部信号のウェイト

【図２６】

Tmset__9,99	タイマーNO 99の内容値を外部タイマー9にセットする
Tmup？__9	外部タイマー9が0か0でないかチェック
Dlyw__65535	タスク固有タイマーに 65535 データを直接セットしてディレイする
Dlywi__ i 10	タスク固有タイマーに i 10 の内容値をセットしてディレイする
Dlywn__99	タスク固有タイマーにタイマーNO 99 の内容値をセットしてディレイする

【図２７】

Ad__i 10, 7	i 10 へ 7ch から A／D 入力する　(0−7)
Da__i 10, 1	i 10 から 1ch へ D／A 出力する　(0,1)
$ 0__c 10, 32	c 10 から 32 バイト送信する。送信完了まで、このアドレスでウェイトする（0CH 用）
$ 1__c 10, 32	c 10 から 32 バイト送信する。送信完了まで、このアドレスでウェイトする（1CH 用）
%0__c 10, 32	受信をウェイトし、受信完了でc 10 から 32 バイトセットする（0CH 用）
%1__c 10, 32	受信をウェイトし、受信完了でc 10 から 32 バイトセットする（1CH 用）

【図２８－１】

i 10＝ i 20＋ i 30 i 10＝ i 20－ i 30 i 10＝ i 20＊ i 30 i 10＝ i 20／ i 30	正整数レジスタ i 同志の四則演算
i 10＝65535 i 10＝ i 20＋65535 i 10＝ i 20－65535 i 10＝ i 20＊65535 i 10＝ i 20／65535	正整数レジスタ i と定数との四則演算
r 10＝ r 20＋ r 30 r 10＝ r 20－ r 30 r 10＝ r 20＊ r 30 r 10＝ r 20／ r 30	実数レジスタ r 同志の四則演算
i 10＝ i 20& i 30	正整数レジスタ i 同志の AND
i 10＝ i 20@ i 30	正整数レジスタ i 同志の OR
i 10＝ i 20& h' f 0 ff	正整数レジスタ i と定数との AND
i 10＝ i 20@ h' 00 ff	正整数レジスタ i と定数との OR
i 10＝ R L(i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回左ローテーション
i 10＝ R R(i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回右ローテーション
i 10＝ S L(i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回左シフト
i 10＝ S R(i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回右シフト
i 10＝ D(i 20)	i 20 の内容を 10 進変換して i 10 にセット
i 10＝ B(i 20)	i 20 の内容を 2 進変換して i 10 にセット
c 10＝ “ABC…	c 10 の先頭から、 c 11、c 12、…と文字 ABC をセット
c 10＝ h' 313233…	c 10 の先頭から、 c 11、c 12、…と 16 進 31、32、33、にセット
c 10＝ i 20# S 5	i 20 の内容を c 10 から 5 文字分ゼロサプレス付きで、c 11、c 12、c 13、c 14、にセット
c 10＝ i 20# Z 5	i 20 の内容を c 10 から 5 文字分ゼロサプレス無しで、c 11、c 12、c 13、c 14、にセット

【図２８－２】

c10＝ｉ20#ｈ	ｉ20の16進内容をそのまま、ｃ10、ｃ11、ｃ12、ｃ13、にセット
c10＝ｒ20#Ｓ4.2	ｒ20の内容をｃ10から整数部４桁、小数部２桁セット、ゼロサプレス付き
c10＝ｒ20#ｚ3.3	ｒ20の内容をｃ10から整数部３桁、小数部３桁セット、ゼロサプレス無し
ｉ10＝ｃ20#5	ｃ20から５文字分をｉ10に２進数としてセット
ｉ10＝ｃ20#ｈ	ｃ20、ｃ21、ｃ22、ｃ23、の16進表示をそのままｉ10にセットする
ｒ10＝ｃ20#6	ｃ20から６文字分をｒ10に浮動小数点セット
ｉ10＝ｒ20	ｒ20整数部４桁をｉ10に２進数としてセット
ｒ10＝ｉ20	ｉ20の５桁（0－65535）をｒ10にセット
ｒ10=SQRT（ｒ20）	平方根
ｒ10＝（ｒ20）＊＊（ｒ30）	ｒ20のｒ30乗
ｒ10=SIN（ｒ20）	ＳＩＮ
ｒ10=COS（ｒ20）	ＣＯＳ
ｒ10=ATN（ｒ20）	ＡＲＣ　ＴＡＮ
ｒ10=LN（ｒ20）	自然対数
ｒ10=LOG（ｒ20）	常用対数

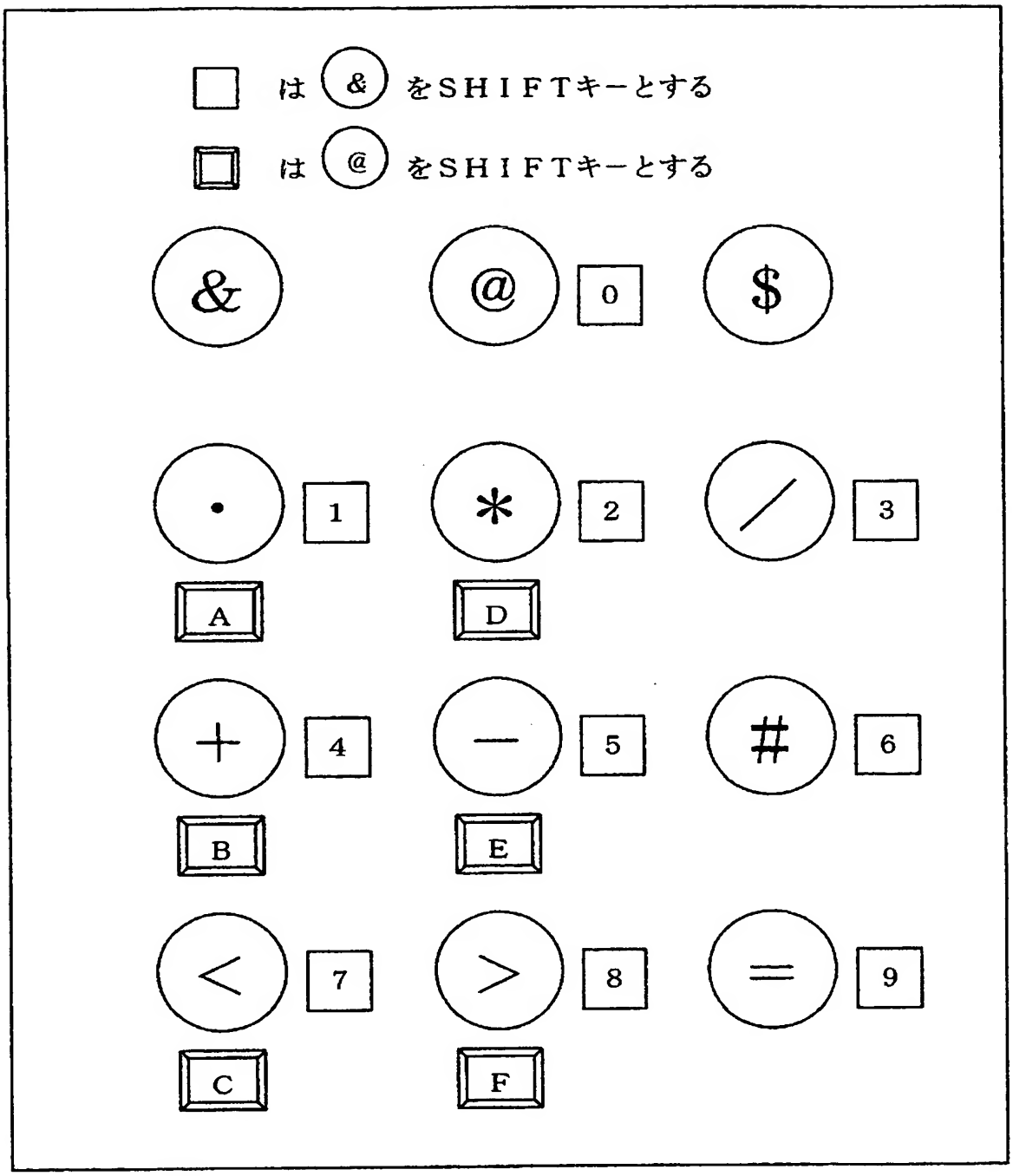
【図２９】

Ｋｙ＿ｉ10	キー入力を即実行し、ｉ10に12ビットセットする
Ｋｙｗ＿ｃ10	キー入力をウェイトし、押されればその変換キーコードをｃ10にセットする
Ｃｈｋ？＿ｃ10,"0,"9…	ｃ10の内容が文字0から9のコード範囲かチェックする。範囲内ならＦａｌｓｅ（0）
Ｄｃｌｒ	画面クリア
Ｄｓｐ＿3,5,"ABC	ｘ＝3，ｙ＝5の位置からＡＢＣ文字表示
Ｄｓｐｃ＿3,5,4,ｃ10	ｘ＝3，ｙ＝5の位置から４文字分ｃ10、ｃ11、ｃ12、ｃ13、の内容の文字表示
Ｄｒｖｓ＿3,5,5	ｘ＝3，ｙ＝5の位置から５文字分反転文字
Ｎｕｍｗ＿3,5,4,ｃ10,ｄ	ｘ＝3，ｙ＝5の位置から0～9キーを入力表示しながら４文字分入力してENTキー又はクリアキー押下までウェイトする。
Ｎｕｍｗ＿3,5,4,ｃ10,ｈ	ｘ＝3，ｙ＝5の位置から0～9,A～Fキーを入力表示しながら４文字入力してENTキー又はクリアキー押下までウェイトする。
Ｒｖｓｓｅｔ＿ｉ10,3,5,4,…	ｘ＝3，ｙ＝5の位置から４文字分,他　ｘ,ｙ,lengthをセットした分、ｉ10をポインターとして、反転をループ状に行う、初期セット。
Ｒｖｓｉｎｃ＿ｉ10,3,5,4,…	上記Rvssetと同じ設定で、この命令を実行する毎に反転がループ状に移行する。

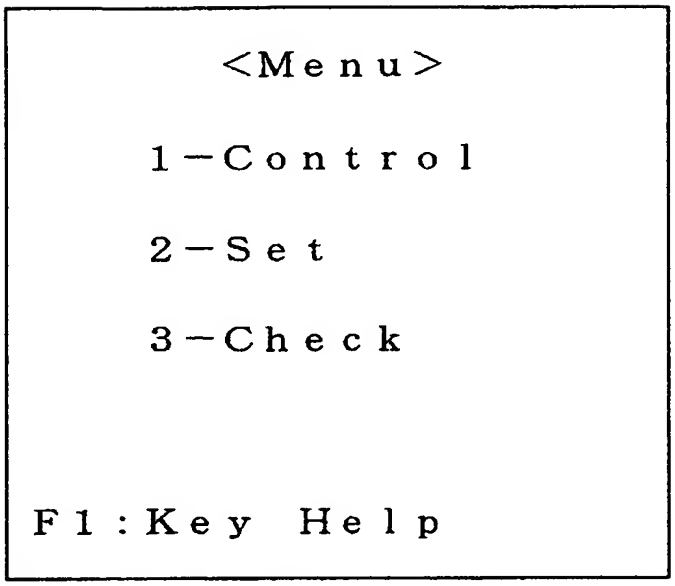
【図３０】

Strani_Vsig, Isig,0 Strani_Vsig, Isig,1 Strano_Osig, Vsig,0 Strano_Osig, Vsig,1	Isigをそのまま Vsigにセット Isigを反転して Vsigにセット Vsigをそのまま Osigにセット Vsigを反転して Osigにセット
Sand_Vsig1,0,Vsig2,0,Vsig3,0	Vsig 2をそのままにしたものと Vsig 3をそのままにしたものと ANDをとってそのままにして Vsig1にセット
Sand_Vsig1,1,Vsig2,1,Vsig3,1	Vsig 2を反転したものと Vsig 3を反転したものと ANDをとって反転して Vsig1にセット
Sor_Vsig1,0,Vsig2,0,Vsig3,0	Vsig 2をそのままにしたものと Vsig 3をそのままにしたものと ORをとってそのままにして Vsig1にセット
Sor_Vsig1,1,Vsig2,1,Vsig3,1	Vsig 2を反転したものと Vsig 3を反転したものと ORをとって反転して Vsig1にセット
Db_AF,CD,56…	バイトテーブル
Dw_3457,AFFF…	ワードテーブル
Getc_Pclabel, c 10,5, i 20	横5バイト毎のラベル Pclabel バイトテーブルから i 20の内容値をインデックスとした、横1列5バイトを c 10から5文字分セットする。
Geti_Pclabel, i 10,5, i 20	横5ワード毎のラベル Pclabel ワードテーブルから i 20の内容値をインデックスとした、横1列5ワードを i 10から5ワード分セットする。
Sfc_c 10, 5, c 20	c 10から5文字分を1文字ずつシフトして先頭 c 10に c 20をセットする。
Sfi_i 10, 5, i 20	i 10から5ワード分、1ワードずつシフトして先頭 i 10に i 20をセットする。
Mv_i 10,Data(i 20)	i 20の内容をインデックスとして Data メモリーから i 10へセットする。
Mv_Data(i 10), i 20	i 10の内容をインデックスとして i 20の内容を Data メモリーへセットする。
Clock_c 10	カレンダークロックから読み出し c 10 を先頭として12文字分、年月日時分秒セット（書き込みはコントローラーで出来る）
;	コメント。CRか；まで。
:	ラベルを識別する。
E	プログラムの終了。

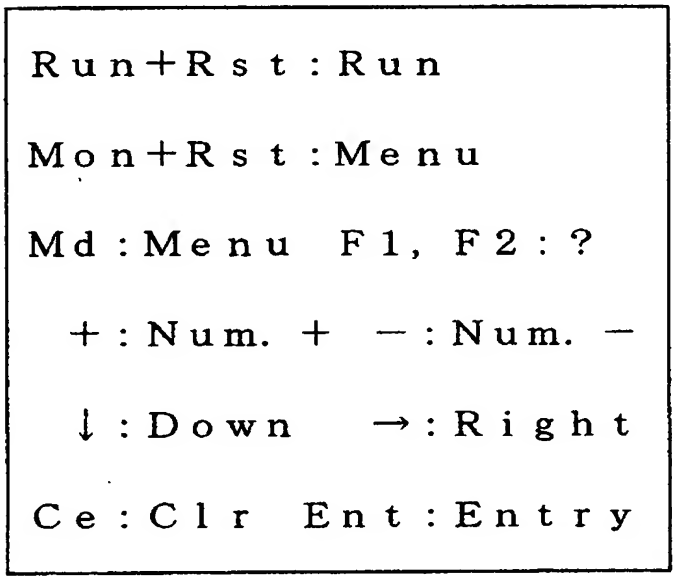
【図３１】



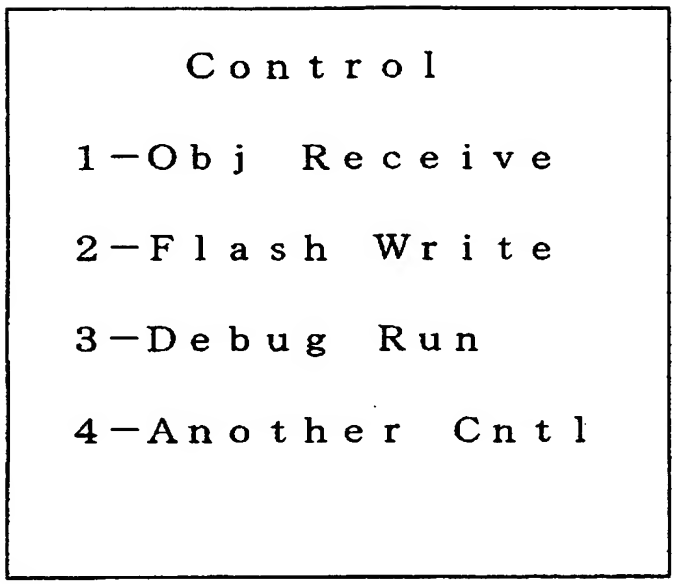
【図３２】


<Menu>
1－Control
2－Set
3－Check
F1:Key Help

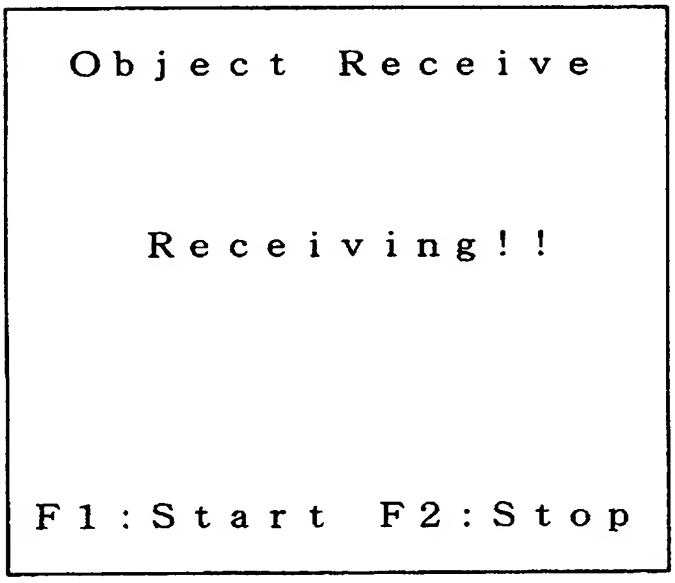
【図３３】


Run+Rst:Run
Mon+Rst:Menu
Md:Menu F1, F2:?
+:Num. + －:Num. －
↓:Down →:Right
Ce:Clr Ent:Entry

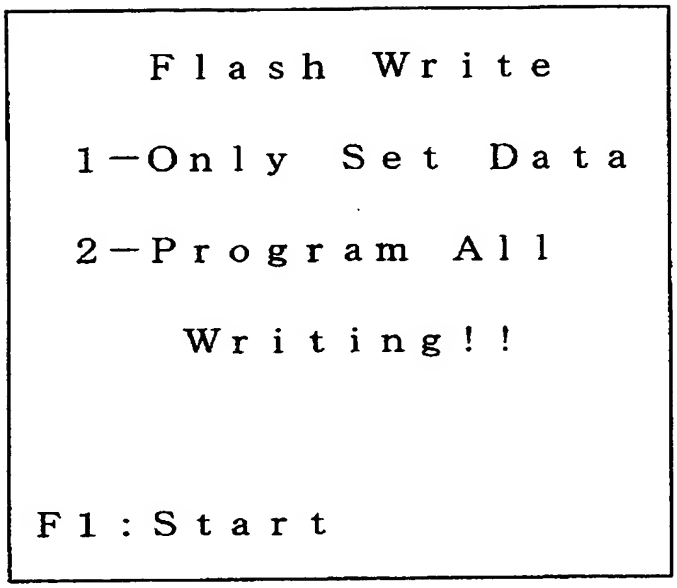
【図３４】


Control
1－Obj Receive
2－Flash Write
3－Debug Run
4－Another Cntl

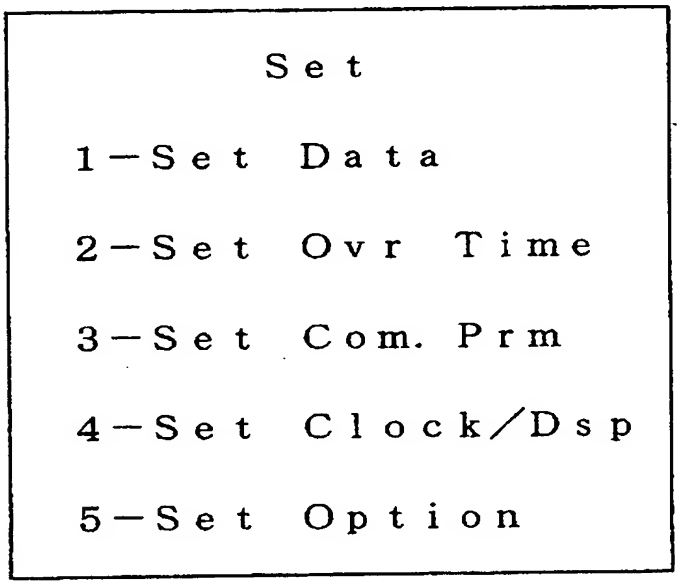
【図３５】


Object Receive
Receiving!!
F1:Start F2:Stop

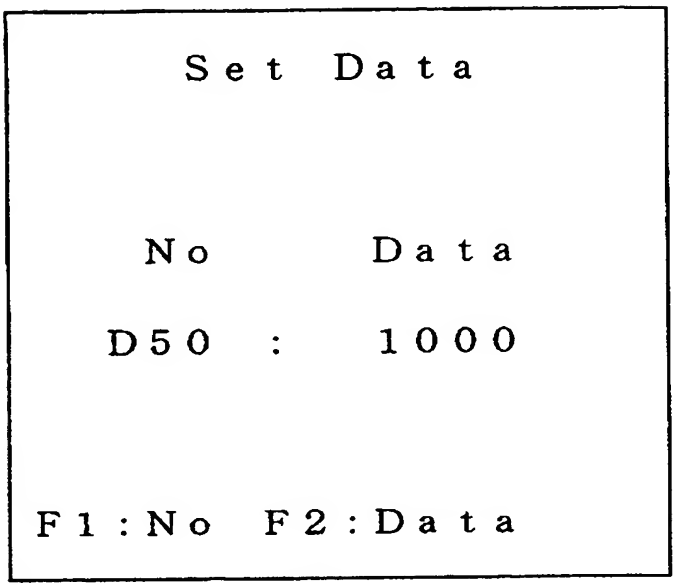
【図３６】


Ｆｌａｓｈ　Ｗｒｉｔｅ
１－Ｏｎｌｙ　Ｓｅｔ　Ｄａｔａ
２－Ｐｒｏｇｒａｍ　Ａｌｌ
Ｗｒｉｔｉｎｇ！！
Ｆ１：Ｓｔａｒｔ

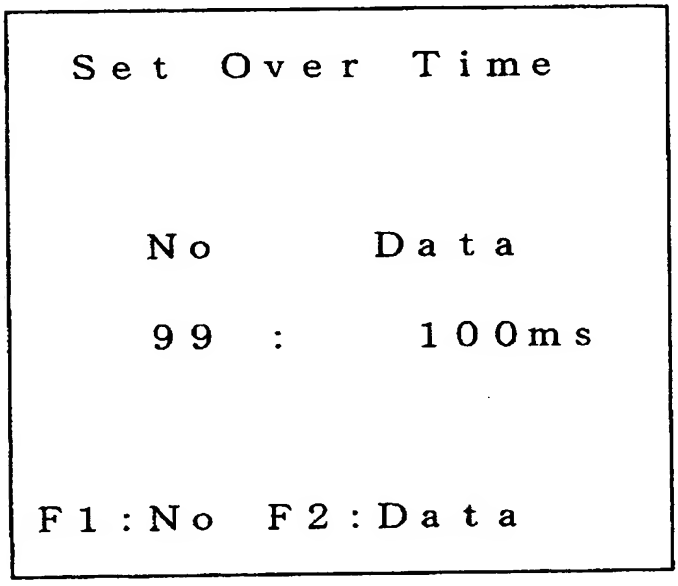
【図３７】


Ｓｅｔ
１－Ｓｅｔ　Ｄａｔａ
２－Ｓｅｔ　Ｏｖｒ　Ｔｉｍｅ
３－Ｓｅｔ　Ｃｏｍ．Ｐｒｍ
４－Ｓｅｔ　Ｃｌｏｃｋ／Ｄｓｐ
５－Ｓｅｔ　Ｏｐｔｉｏｎ

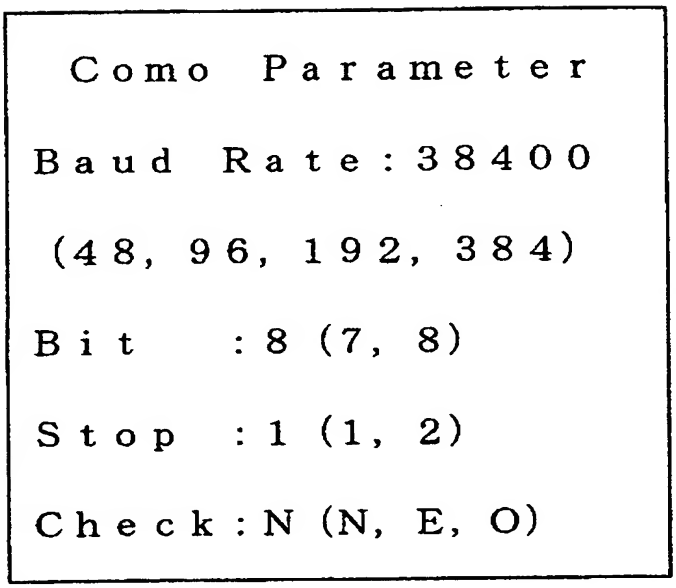
【図３８】


Set Data
No Data
D50 : 1000
F1:No F2:Data

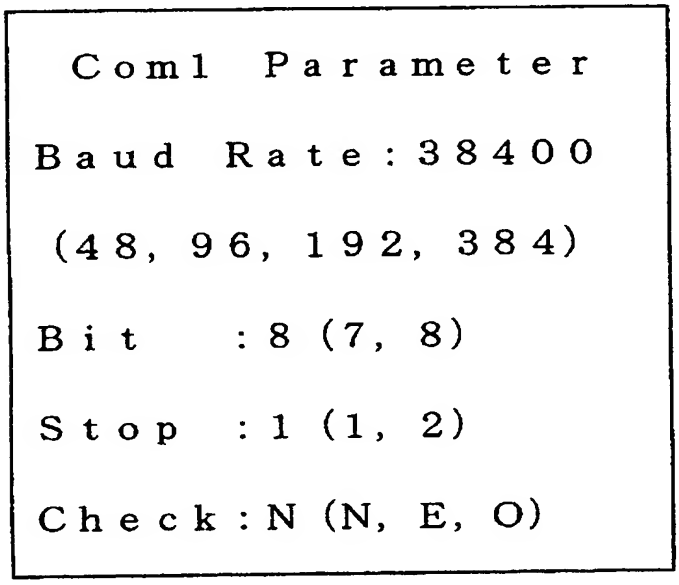
【図３９】


Set Over Time
No Data
99 : 100ms
F1:No F2:Data

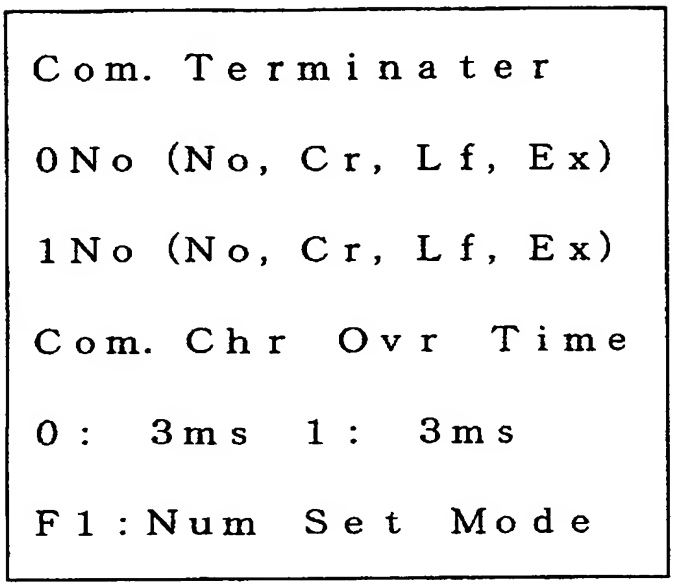
【図４０】


Como Parameter
Baud Rate:38400
(48, 96, 192, 384)
Bit :8 (7, 8)
Stop :1 (1, 2)
Check:N (N, E, O)

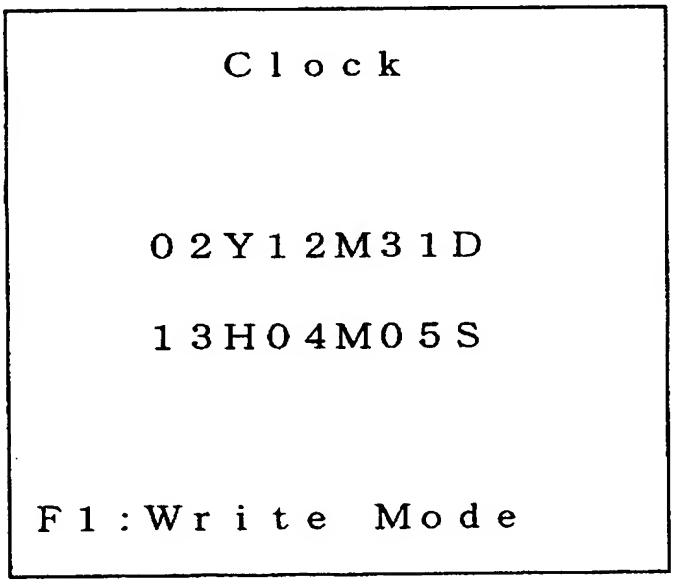
【図４１】


Com1 Parameter
Baud Rate:38400
(48, 96, 192, 384)
Bit :8 (7, 8)
Stop :1 (1, 2)
Check:N (N, E, O)

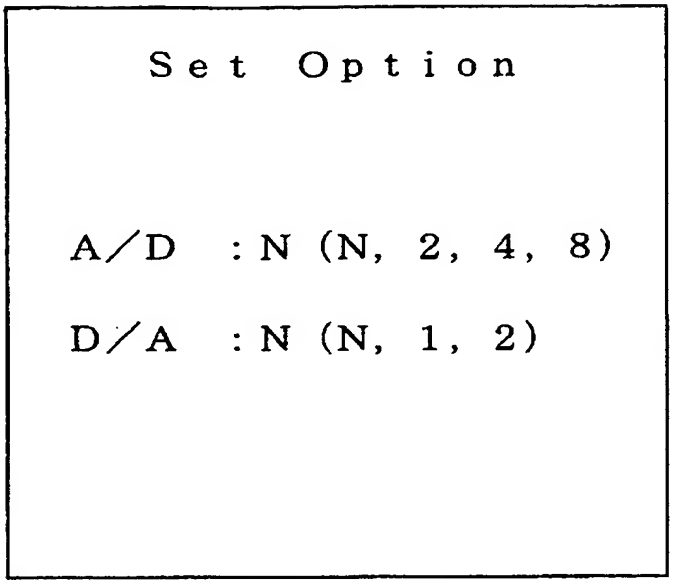
【図４２】


Com. Terminater
0No (No, Cr, Lf, Ex)
1No (No, Cr, Lf, Ex)
Com. Chr Ovr Time
0: 3ms 1: 3ms
F1:Num Set Mode

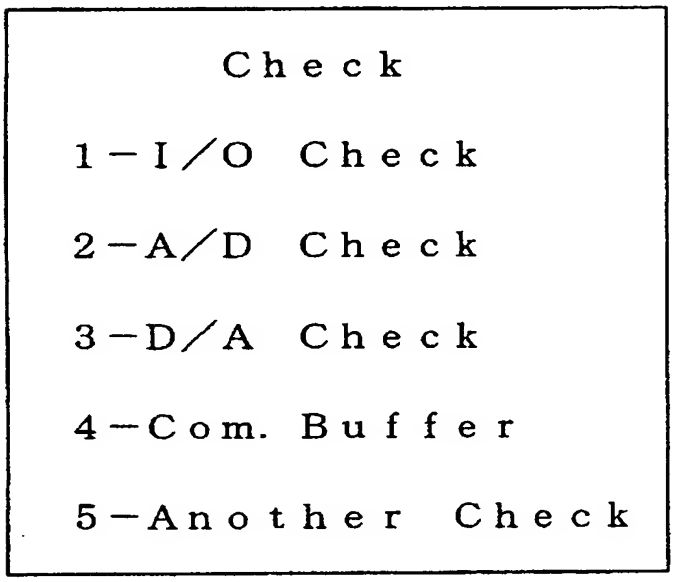
【図４３】


Clock
02Y12M31D
13H04M05S
F1:Write Mode

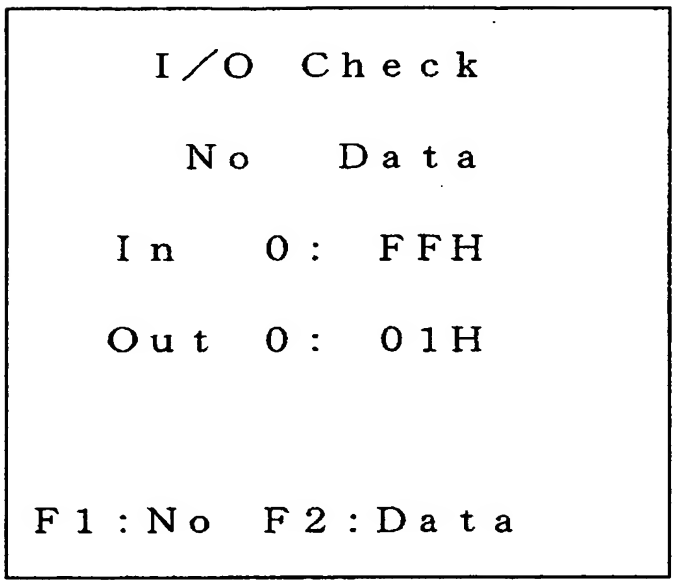
【図４４】


Set Option
A/D :N (N, 2, 4, 8)
D/A :N (N, 1, 2)

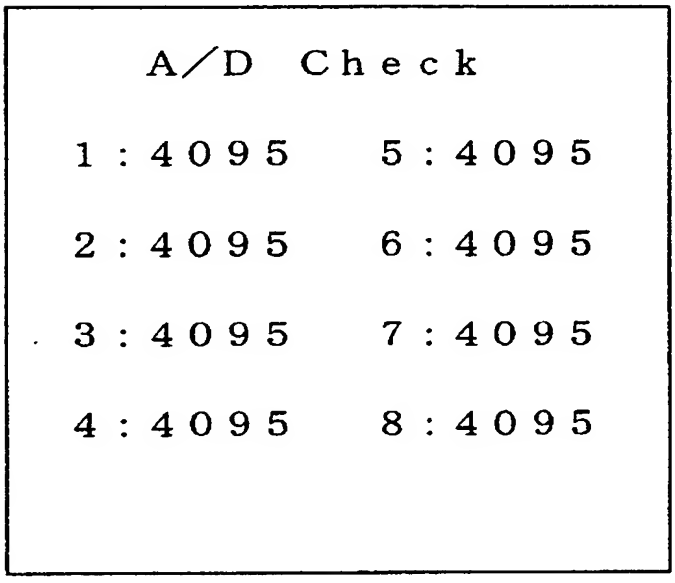
【図４５】


Check
1－I/O Check
2－A/D Check
3－D/A Check
4－Com. Buffer
5－Another Check

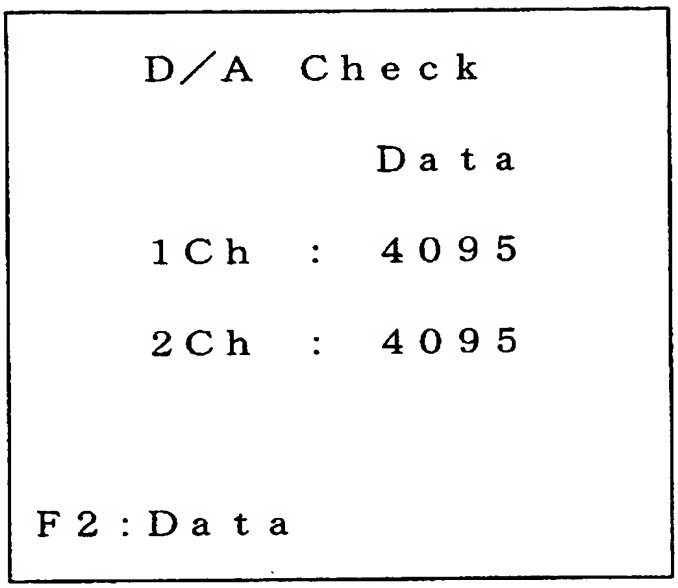
【図４６】


I／O Check
No Data
In 0: FFH
Out 0: 01H
F1:No F2:Data

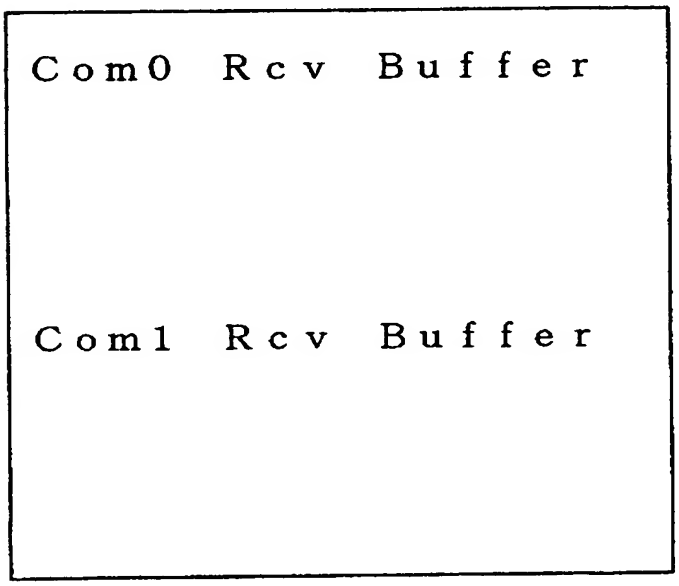
【図４７】


A／D Check
1:4095 5:4095
2:4095 6:4095
3:4095 7:4095
4:4095 8:4095

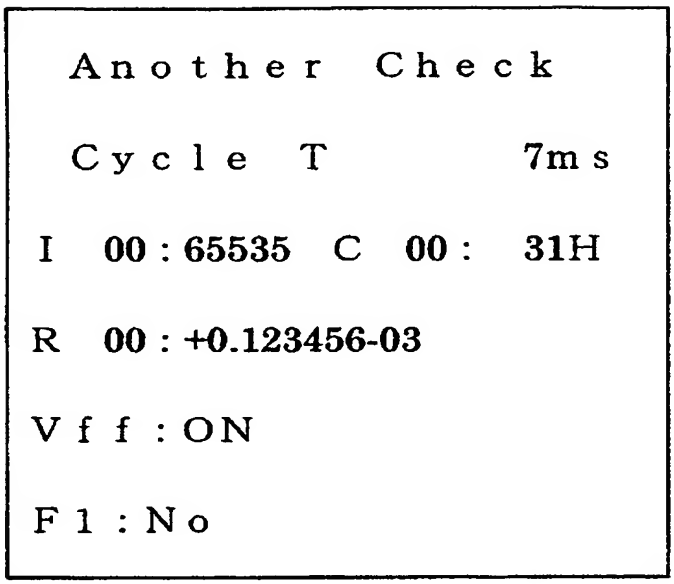
【図４８】


D／A Check
Data
1Ch : 4095
2Ch : 4095
F2：Data

【図４９】


Com0 Rcv Buffer
Com1 Rcv Buffer

【図５０】


Another Check
Cycle T 7ms
I 00:65535 C 00: 31H
R 00:+0.123456-03
Vff:ON
F1:No

【書類名】　　要約書

【要約】

【課題】

コンピューター技術、特に産業機器は特異な言語を駆使するため技術者には一般大衆と隔絶した砦の様なものが形成しており、時には、それが弊害となり、意思の疎通が感じられる事もあり、しばしばその横暴ぶりに痛感する事がある。

制作金額、納期について、我を通し、簡単な変更の発生時にも価格的に高目になったり、しばしばユーザーの嘆きを聞く事になる。又、プログラマーとして起業した初心者に関して、第１回の開発でつまずき、二度と信頼を得る事がなく、挫折の浮き目に会った多くの人々を見てきた。

本、発明はその様な砦を壊し、誰でも、気安くＦＡに従事して行ける様、一念発起して制作したものである。

【解決手段】

多重処理を意識させずに、流れ図（フローチャート）にそった形で、しかも、プログラム制作に指針を与えながら選択式にして、コーディング作業などの開発工数を省く。パッケージ化する事により、バグ発生をさせないようにする。

認定・付加情報

特許出願の番号	特願２００２－３８３０８３
受付番号	２０２０２３３０３８７
書類名	特許願
担当官	塩野　実　　２１５１
作成日	平成１５年　４月　８日

＜認定情報・付加情報＞

【提出日】	平成14年12月 6日
【特許出願人】	申請人
【識別番号】	503029670
【住所又は居所】	大阪府大阪市鶴見区鶴見５丁目６番９号３０１室
【氏名又は名称】	加藤　宝一

次頁無

特願２００２－３８３０８３

出　願　人　履　歴　情　報

識別番号　　［５０３０２９６７０］

１．変更年月日　　２００２年１２月　６日
［変更理由］　　新規登録
住　所　　大阪府大阪市鶴見区鶴見５丁目６番９号３０１室
氏　名　　加藤　宝一